UIL Computer Science Competition

Region 2022

JUDGES PACKET - CONFIDENTIAL

I. Instructions

1. The attached printouts of the judge test data are provided for the reference of the contest director and programming judges. Additional copies may be made if needed for this purpose.

2. This packet must remain CONFIDENTIAL. Additional copies may be made and returned to schools when other confidential contest material is returned.

II. Table of Contents

Problem 1
60 Points

1. Agustina

Program Name: Agustina.java **Input File: None**

Test Output to Screen:
```
1A-Slidell HS
2A-San Augustine HS
3A-Fairfield HS
4A-Dallas TAG
5A-Lucas Lovejoy HS
6A-Cypress Woods HS
```

Problem #2
60 Points

2. Arya

Program Name: Arya.java **Input File: arya.dat**

Test Input File:

```
8
10 7
15 12
100 32
1000 500
31 31
1 1
50 49
50 1
```

Test Output to Screen:

```
1-5-10
6-8-10
6-6-7
7-7-7
GOT IT!!!
1-8-15
9-12-15
GOT IT!!!
1-50-100
1-25-49
26-37-49
26-31-36
32-34-36
32-32-33
GOT IT!!!
1-500-1000
GOT IT!!!
1-16-31
17-24-31
25-28-31
29-30-31
31-31-31
GOT IT!!!
1-1-1
GOT IT!!!
1-25-50
26-38-50
39-44-50
45-47-50
48-49-50
GOT IT!!!
1-25-50
1-12-24
1-6-11
1-3-5
1-1-2
GOT IT!!!
```

Problem #3
60 Points

3. Diego

Program Name: Diego.java **Input File: diego.dat**

Test Input File:

```
10
2 4
02 1 0
34 0 0
56 0 0
78 0 0
5 2
12345 0 0
67890 0 0
2 9
57 0 0
15 0 0
77 0 0
37 0 0
02 0 0
26 1 1
22 0 0
64 1 1
28 0 0
5 4
98842 2 1
52033 2 1
65406 1 1
55312 1 1
4 5
8307 1 0
9684 2 0
3852 1 0
4587 2 1
3233 0 0
6 4
389241 3 2
182070 2 0
659434 3 0
327772 1 1
2 9
06 0 0
73 1 0
08 1 1
59 0 0
46 0 0
31 1 1
26 0 0
08 1 1
76 0 0
5 1
18500 0 0
2 4
86 1 0
80 0 0
91 0 0
78 0 0
4 5
0326 1 0
1656 0 0
3386 2 1
9006 1 0
3179 3 2
```

Test Output to Screen:

```
4
0
1
15
12
250
1
7776
4
3
```

Problem #4
60 Points

4. Fai

Program Name: Fai.java **Input File: fai.dat**

Test Input File:

```
8
A,B,C,D,E,F,G,H
C
A<->B,A<->C,B<->C,B<->D,B<->F,C<->E,C<->G,E<->G,G<->F,G<->H,H<->F,F<->D
--------------------
A,B,C,D,E,F,G,H
C
A<->B,A<->C,B<->C,B<->D,B<->F,C<->E,C<->G,E<->G,G<->F,H<->F,F<->D
--------------------
A,B,C,D,E,F,G,H
D
A<->B,A<->C,C<->B,C<->D,C<->E,D<->F,E<->F,F<->G,F<->H,G<->H
--------------------
A,B,C,D,E,F,G,H
D
A<->C,C<->B,C<->D,C<->E,D<->F,E<->F,F<->G,F<->H,G<->H
--------------------
A,B,C,D,E
E
A<->B,B<->C,C<->D,E<->D,A<->E
--------------------
A,B,C,D,E
E
A<->B,B<->C,C<->D,E<->D,A<->E,E<->B
--------------------
A,B,C,D,E
A
A<->B,B<->C,C<->D,E<->D,A<->E,A<->C,B<->E,C<->E,D<->B,D<->A
--------------------
A,B,C,D,E
A
A<->B,B<->C,C<->D,E<->D,A<->E,A<->C
--------------------
```

Test Output to Screen:

```
Test case 1: possible
Test case 2: impossible
Test case 3: possible
Test case 4: impossible
Test case 5: possible
Test case 6: impossible
Test case 7: possible
Test case 8: impossible
```

Problem #5
60 Points

5. Ivan

Program Name: Ivan.java **Input File: ivan.dat**

Test Input File:

```
15
BBEEBCECDCCCCDDBDAEBBBECAEBCBBECAEBCDEED
BBEEBCECDCCCCDDBDAEBBBECAEBCBBECAEBCDEED
BCEEBAECDCDCCDDADAEBEB_CAEBCB_ECAE_CD___
AB_DEA_CDEAB_DEABCD_ABCDEAB_DEABC_EABCD_
BACBEDACBACBEDAAECBDBADCDBBAACCDABECBAAD
________________________________________
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
BEEBCECDCCCCDDBDAEBBBECAEBCBBECAEBCDEEDD
CBEABCEAB_B_DDEE_D_CE_E_ECB_ABEBA_DDBEEB
_BEEBCEADCC_CDB_DAEBBBECAEBABBECAEACD_EC
ABEEECECDCCCCDDBDAEABEECAEBCABECAEECDEED
CCAACDADEDDDDEECEBACCCADBACDCCADBACDEAAE
```

Test Output To Screen:

```
Exam #1: 240 100.0
Exam #2: 164 85.3
Exam #3: -18 18.2
Exam #4: 0 25.0
Exam #5: 0 0.0
Exam #6: -56 7.5
Exam #7: 8 27.5
Exam #8: 0 25.0
Exam #9: -32 15.0
Exam #10: 0 25.0
Exam #11: 8 27.5
Exam #12: 40 40.6
Exam #13: 176 86.1
Exam #14: 192 85.0
Exam #15: -80 0.0
```

Problem #6
60 Points

6. Juana

Program Name: Juana.java **Input File: juana.dat**

Test Input File:

```
10
4 5
    1     2     3     4     5
    6     7     8     9    10
   11    12    13    14    15
   16    17    18    19    20
3 2 3 2
7 6
    1     2     3     4     5     6
    7     8     9    10    11    12
   13    14    15    16    17    18
   19    20    21    22    23    24
   25    26    27    28    29    30
   31    32    33    34    35    36
   37    38    39    40    41    42
5 4 2 3
3 3
0 0 0
0 0 0
0 0 0
2 2 3 1
15 15
  380  818  120  401  830  484  494  145  557  725  945  226  110  979  955
  127  503  672  162   84  469  695  139  251  150  487  528  461  137  569
  748  538   21  397  303  403  128  110  443  952  339  262  588   64  442
  887  847  113  998  150  302   86  492  277   19  395  411  294  677  205
  214  382  204  798  925  986  451  520  319   19  606  754  545 1000   51
  259  738  134  707  183  502  795  921  434   28  916  882  163  320  531
  366  557  668  616   75  172  459  528  691  212  948  768   56  574  490
  225  180  488  332  650   31  368  911  738  547  204  577  472  409  113
  454  700  493  620  837  683  753  727  822  482  110  333  268  437   35
  104  816  591  178  886  522   35  955  580  450  979  805  588  803  856
   69  303  197  245  461  122  806  161  687  225  255  385  600  910  171
  352  160  507  600  916   89  529  613  815  121   37  753  754  597   15
  211  798  545  916  356  285  648  535  691  112   87  608  481  613  198
  323  768  515  562  179   83  191  107  430  545  430  651  179  330  132
  808  751  745  628  746  558   62  991  127  665  101  936  679  265  410
3 1 15 13
2 2
  159  802
  287  140
1 1 2 2
2 15
  904  507  370  522  445  287  109  674  415  935  287  936  692  231  853
  437  816  734  141  735  323  674  420  550  919  605  970   91   72  889
1 5 5 2
```

~ Juana, input continues next page ~

~ Juana, input continued ~

```
15 2
  215  286
    4  189
  338    0
  424  744
  941  798
  823  232
  890  700
   31  159
  253  818
  394  728
  290  808
  692  249
  664  260
  567  341
  566   18
5 1 2 7
12 12
  914  816  721  239  938  485   54  132  180  621  553  406
  459   55  634  138  635  562   32  683  930  738   84  512
  969  481  329  915  311  829  681   71  843   47  274  967
  218   84  612  308  191  569  558  677  337  137  283  887
  155  413  257  413  963  742  510  230  901  514  692  164
  887  748  532  360   12  337  538  113  607  907  156  439
  605  905  603  578  895  454  866  852  618   14  344  618
  886  131    3  514  181  887  941  297  765  581   68   63
  475  104  783  495  198  206  226  860  568  529  670  844
  193   76   49  564  813  896  948  816  241  959  767  492
  390  236  183  787  882  157  229  637  295  616  282   54
  711  416  890  594  349  858  990  416  425  909  259  361
4 8 3 5
5 3
  958   27  984
  402  135  213
  520   11  371
  769  608  750
  291  525  791
5 2 2 2
2 2
  795  656
  260  630
1 1 2 3
```

~ Juana, output on next page ~

~ Juana, continued ~

Test Output To Screen:

```
Test case #1:
   12   13   14
   17   18   19
--------------------
Test case #2:
   28   29
   34   35
   40   41
--------------------
Test case #3: Unable to extract requested size!
--------------------
Test case #4:
  748  538   21  397  303  403  128  110  443  952  339  262  588   64  442
  887  847  113  998  150  302   86  492  277   19  395  411  294  677  205
  214  382  204  798  925  986  451  520  319   19  606  754  545 1000   51
  259  738  134  707  183  502  795  921  434   28  916  882  163  320  531
  366  557  668  616   75  172  459  528  691  212  948  768   56  574  490
  225  180  488  332  650   31  368  911  738  547  204  577  472  409  113
  454  700  493  620  837  683  753  727  822  482  110  333  268  437   35
  104  816  591  178  886  522   35  955  580  450  979  805  588  803  856
   69  303  197  245  461  122  806  161  687  225  255  385  600  910  171
  352  160  507  600  916   89  529  613  815  121   37  753  754  597   15
  211  798  545  916  356  285  648  535  691  112   87  608  481  613  198
  323  768  515  562  179   83  191  107  430  545  430  651  179  330  132
  808  751  745  628  746  558   62  991  127  665  101  936  679  265  410
--------------------
Test case #5:
  159  802
  287  140
--------------------
Test case #6:
  445  287  109  674  415
  735  323  674  420  550
--------------------
Test case #7:
  941  798
  823  232
  890  700
   31  159
  253  818
  394  728
  290  808
--------------------
Test case #8:
  677  337  137
  230  901  514
  113  607  907
  852  618   14
  297  765  581
--------------------
Test case #9: Unable to extract requested size!
--------------------
Test case #10: Unable to extract requested size!
--------------------
```

Problem #7
60 Points

7. Krithika

Program Name: Krithika.java **Input File: krithika.dat**

Test Input File: (Continuation lines are indented below the initial line)

```
20
3 2
14 159 26
4 3
1 1 1 2
5 4
1 2 3 4 5
2 1
999999999999999999 1
2 2
999999999999999999 1
3 1
1 999999999999999999 1
3 2
1 999999999999999999 1
5 1
1 2 4 8 16
5 2
1 2 4 8 16
1 1
1
1 1
50
1 1
1000000000000000000
50 37
31 8 35 2 13 1 27 31 1 44 35 45 47 41 18 48 11 39 44 19 47 38 33 6 12 46 34 38 13 24 33 12 29 9 45 27 45 7 26 1 34
    45 19 15 20 49 21 48 42 39
50 11
12 34 11 26 22 6 34 45 38 49 44 6 16 26 43 22 4 8 28 36 44 49 28 6 42 12 31 21 47 18 18 46 38 15 32 32 45 16 5 32
    6 27 34 9 28 33 11 44 6 43
50 39
14 37 28 46 17 46 12 13 46 43 28 45 45 23 16 26 35 12 17 30 11 13 26 39 45 37 34 17 40 21 45 22 18 47 9 5 6 21 15
    4 48 25 1 12 19 17 33 7 43 3
1000 192
315 263 457 374 606 673 666 111 84 551 686 962 239 832 939 453 854 884 562 874 673 363 112 343 289 107 312 258 652
    923 381 812 70 847 370 11 753 234 354 301 932 155 348 814 402 318 851 742 479 859 786 8 425 320 693 157 965
    874 927 414 689 831 292 846 597 509 110 594 253 555 427 773 375 148 537 501 924 326 756 927 307 544 999 70 808
    875 619 130 362 52 536 310 52 827 784 724 832 861 215 616 954 246 278 174 463 345 665 909 677 658 373 364 184
    887 287 375 577 781 779 920 581 57 926 165 93 604 58 498 79 923 832 340 658 179 549 857 992 470 761 568 608 4
    660 109 875 147 852 784 937 246 288 448 349 682 773 335 462 817 595 510 584 557 446 893 19 939 574 260 284 821
    277 374 355 680 86 169 495 421 351 366 691 13 917 693 961 790 531 665 806 618 539 363 270 974 270 725 7 719
    433 911 353 13 911 15 94 12 883 610 543 656 595 702 102 216 490 419 472 109 376 52 796 664 444 689 697 4 775
    613 670 375 435 199 287 337 53 201 683 817 42 361 842 313 605 316 642 333 458 599 159 67 697 721 545 442 246
    828 464 549 893 374 582 118 89 153 629 184 42 587 519 684 711 256 513 982 806 575 598 999 954 864 758 348 144
    699 830 327 44 286 712 180 923 286 893 879 936 690 623 352 994 504 918 937 506 39 856 301 440 615 944 232 982
    615 517 163 191 733 765 349 733 333 141 946 565 633 55 132 577 29 771 285 978 506 211 324 219 311 548 269 174
    898 151 697 320 464 661 610 587 839 802 858 105 864 21 983 5 758 599 316 213 132 296 454 109 889 517 742 240
    77 161 787 610 50 717 31 250 5 335 149 921 182 811 266 403 273 22 134 411 832 87 928 447 999 427 634 706 509
    681 920 866 724 165 963 435 185 289 32 821 644 854 818 826 922 134 37 701 157 48 587 327 753 217 521 65 379
    214 40 232 592 73 860 129 165 332 288 421 305 798 304 125 162 497 155 60 946 25 230 494 77 79 549 758 137 424
    990 690 147 699 823 737 264 866 108 978 298 673 970 187 625 113 509 21 902 713 435 442 532 211 787 7 238 973
    809 460 563 400 588 360 90 674 894 994 153 103 84 304 412 325 773 980 755 257 586 139 972 376 996 722 327 7
    787 308 989 492 509 280 106 380 951 766 47 741 647 963 843 6 661 587 345 807 210 144 730 148 270 740 461 187
    441 396 937 410 760 28 49 616 144 644 470 840 117 887 777 496 964 363 581 425 434 755 110 263 186 805 66 796
    452 863 98 465 74 973 287 477 587 523 769 462 72 103 461 632 408 755 597 440 322 234 198 166 982 278 96 30 685
    535 162 332 675 633 867 114 445 104 733 288 262 539 840 654 728 302 899 112 61 398 990 404 329 706 102 830 787
    318 928 194 987 999 109 391 907 848 954 670 90 522 318 431 941 137 602 767 619 707 606 283 720 653 3 850 169
    754 226 944 335 896 659 565 208 868 695 940 68 568 442 260 823 215 692 193 97 345 842 758 964 54 824 344 38
    893 904 73 280 371 176 309 34 91 210 342 333 403 696 231 660 136 232 493 476 12 553 174 637 187 168 571 913
    606 469 133 260 287 807 988 388 607 450 881 615 569 4 986 219 791 460 15 903 875 515 429 251 217 593 970 62 77
    922 483 562 451 407 153 748 431 536 615 813 82 330 429 348 115 411 474 808 764 872 781 782 392 503 223 194 18
    108 759 410 720 693 957 303 268 790 517 966 871 606 258 151 225 138 64 469 913 494 16 855 626 489 19 832 451
    584 948 883 670 320 234 773 812 268 859 571 929 913 469 186 641 280 513 196 37 856 868 48 886 854 704 482 210
    478 794 795 731 532 333 540 651 242 658 827 230 985 898 253 364 548 325 704 335 108 102 79 928 949 46 520 946
    632 426 596 574 569 418 414 38 116 858 386 215 605 549 396 25 55 850 888 28 691 902 462 823 635 368 396 714
    739 715 597 76 82 471 374 381 419 544 610 140 991 433 539 191 99 538 447 437 504 600 742 180 79 379 210 284
    756 568 690 519 746 673 421 704 42 440 952 999 477 107 800 696 297 475 687 560 394 818 663 403 636 371 164 939
    569 219 413 938 72 306 354 158 192 505 732 616 578 256 474 909 501 31 565 694 661 661 272 922 894 909 584 491
    468 10 612 148 382 200 643 774 125 296 961 577 433 741 431 145 278 489 100 743 797 534 342 877 201 409 819 556
    711 118 550 910 483 264 460 750 353 613 667 510 467
1000 111
680 212 629 907 863 729 672 519 770 725 162 513 60 357 585 738 958 27 961 296 283 928 496 676 346 3 166 667 704
    767 669 27 193 417 265 179 986 944 210 667 246 894 485 340 716 853 748 615 447 601 801 126 188 892 92 461 879
    411 55 449 70 695 954 805 614 251 945 11 190 170 913 135 987 898 113 512 793 775 94 136 206 111 992 751 536
    383 469 868 960 467 663 76 396 548 8 65 909 862 422 510 862 998 977 995 996 622 70 995 183 961 627 683 692 556
    964 885 882 253 753 201 752 48 230 963 453 21 181 5 773 619 268 894 863 388 695 981 437 412 125 978 972 772
    258 630 532 115 326 235 632 625 483 776 657 348 76 982 196 510 752 804 665 390 584 455 497 948 71 399 125 728
    409 898 108 413 628 329 373 915 609 944 494 394 436 630 537 226 82 172 747 505 69 38 287 633 340 761 711 686
    952 788 924 403 801 398 237 755 290 607 8 752 996 757 23 981 735 840 987 82 113 440 676 581 278 475 371 972
    882 249 304 223 490 876 308 280 976 915 270 456 793 396 35 832 695 151 505 26 363 806 685 185 407 897 353 268
    92 48 781 969 355 498 95 110 347 787 348 760 938 647 858 791 133 195 941 798 991 347 964 400 369 649 34 966
    372 821 922 18 629 545 625 471 23 928 728 244 399 638 795 402 892 181 977 636 440 127 53 482 416 715 418 254
    576 73 482 587 87 521 175 493 527 373 741 199 173 965 559 150 944 686 107 421 543 930 42 436 902 174 22 86 77
    438 529 584 862 984 413 968 206 470 202 411 781 838 253 819 369 263 700 917 35 637 716 316 424 687 471 476 930
    602 365 613 599 66 995 90 834 767 156 799 140 993 970 193 523 632 87 60 388 31 288 693 23 93 203 739 697 61
    582 116 129 483 853 671 114 867 357 295 180 859 356 132 710 346 889 293 617 959 579 619 378 359 703 470 738
    462 549 542 40 101 509 630 250 281 581 268 95 793 600 223 986 941 15 870 695 336 711 814 589 276 988 252 555
    310 318 255 103 588 871 470 196 614 640 64 18 903 938 357 908 121 612 247 192 65 19 461 780 968 419 879 79 197
    819 911 593 270 425 757 546 151 886 73 922 364 746 495 29 681 587 775 203 845 706 936 107 280 242 264 29 958
    26 898 536 941 342 778 321 752 751 42 881 429 475 909 472 912 865 180 866 917 607 802 292 587 671 166 310 415
    569 275 50 955 982 650 814 351 453 326 835 633 887 95 803 875 716 372 362 268 643 652 487 53 957 553 831 86
    162 573 182 63 271 502 174 482 36 931 858 624 946 934 694 746 956 399 346 909 94 847 975 15 773 452 374 977
    696 241 624 443 991 79 285 783 431 900 384 117 694 94 525 201 623 784 446 824 354 677 92 680 904 863 371 225
    423 44 183 498 918 626 653 1 205 323 772 6 993 41 992 959 2 846 442 439 372 326 697 967 976 129 981 617 696
    240 30 447 374 467 304 684 148 242 608 829 557 74 351 571 374 996 746 363 860 549 164 52 921 986 735 991 927
    76 902 206 124 969 163 73 137 482 978 899 50 413 517 847 33 744 877 501 284 709 498 573 768 77 396 257 984 485
    346 109 945 626 806 356 726 117 119 34 571 204 259 182 125 539 318 472 618 151 770 379 833 915 865 154 844 346
    921 722 594 534 612 762 189 900 801 872 472 601 459 747 33 846 47 653 174 564 424 118 653 717 389 9 919 977
    799 519 753 861 36 683 837 241 845 93 573 939 597 778 665 8 154 458 435 41 360 772 381 914 423 156 72 879 804
    787 647 981 619 597 906 496 854 188 330 934 98 852 522 914 22 896 338 640 566 98 356 451 762 199 869 266 97
    333 22 384 393 56 907 43 535 12 839 169 587 187 52 544 535 387 717 129 681 351 794 331 407 513 473 751 435 101
    878 38 711 144 12 919 297 819 249 548 539 907 292 731 911 995 278 315 42 165 5 734 780 695 623 361 823 653 733
    680 208 779 318 833 338 913 501 266 132 852 844 361 340 343 132 271 975 642 918 707 928 677 291 627 153 434
    450 914 785 94 261 146 995 743 64 898 158 868 572 755 599 300 188 583 108 572 386 250 659 45 109 429 899 585
    829 978 290 821 846 469 418 250 50 427 363 455 12 574 784 592 815 383 330 126 908 900 940 96 689 928 179 561
    880 90 482 663 263 445 770 225 464 406 436 130 815 607 967 120 220 202 14 266 324 75 699 671 130 681 635 416
    730 482 615 333 692 224 676 798 349 262 780 371
50 10
513456986490769832 860365840542438967 739622663538751597 631936686173957517 357153625078220040 384314386664353650
    985923013829181413 389795504167498262 783986615723912665 810793556037049088 674886340026209680
    285227627159961607 907624635140524145 333871945962276296 611737967894988660 152539665109350155
    958169428336171082 305871750658073332 853227283253034839 757334282336987311 883932772881382213
    526714595868838075 678197221339803669 395186602621172742 649894542492136983 222493282092333997
    493594786849366465 191931675595203981 793645612421856718 674165261428700996 632246463153433516
    392686885241410012 671192344799917759 712362442316735431 384396050490380546 165077269115499001
    214933794822375725 189986291977555493 716123973576531 47 869601572137703161 705998793296452555
    779896142245246526 894504541603682381 921314671324742379 284101145361357653 876739576372846194
    214744127327602858 221874818129749894 966816076148427527 406171370096700058
1000 1000
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 1 1 1 1 1 1 1 1
1000 1000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
    1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
```

```
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
```

```
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
```

```
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000 1000000000000000000
```

Test Output To Screen:

```
Case #1: 26
Case #2: 1
Case #3: 0
Case #4: 999999999999999999
Case #5: 1
Case #6: 999999999999999999
Case #7: 1
Case #8: 16
Case #9: 0
Case #10: 1
Case #11: 50
Case #12: 1000000000000000000
Case #13: 0
Case #14: 32
Case #15: 0
Case #16: 768
Case #17: 896
Case #18: 666532744850833408
Case #19: 1
Case #20: 1000000000000000000
```

Problem #8
60 Points

8. Michal

Program Name: Michal.java **Input File: michal.dat**

Test Input File:
```
20
01234567899876543210
0
1
2
3
4
5
6
7
8
9
0794856123
222222
3333333
1086420
1111111111111
88888
4749375973597
232349834897394893
958473728589694843 7
```

Test Output To Screen:
```
**************************************************************
* _     _  _     _  _  _  _  _  _  _  _  _  _     _  _     _ *
*| |  | _| _||_||_ |_   ||_||_||_||_|  ||_ |_ |_| _| _|  || |*
*|_|  ||_  _|  | _||_|  ||_|  |  ||_|  ||_| _|  | _||_   ||_|*
*                                                            *
**************************************************************
*****
* _ *
*| |*
*|_|*
*   *
*****
*****
*   *
*  |*
*  |*
*   *
*****
*****
* _ *
* _|*
*|_ *
*   *
*****
*****
```

```
* _ *
* _|*
* _|*
*   *
*****
*****
*   *
*|_|*
*  |*
*   *
*****
*****
* _ *
*|_ *
* _|*
*   *
*****
*****
* _ *
*|_ *
*|_|*
*   *
*****
*****
* _ *
*  |*
*  |*
*   *
*****
*****
* _ *
*|_|*
*|_|*
*   *
*****
*****
* _ *
*|_|*
*  |*
*   *
*****
********************************
* _  _  _     _  _  _     _  _ *
*| |  ||_||_||_||_ |_   | _| _|*
*|_|  |  |  ||_| _||_|  ||_  _|*
*                              *
********************************
********************
* _  _  _  _  _  _ *
* _| _| _| _| _| _|*
*|_ |_ |_ |_ |_ |_ *
*                  *
********************
***********************
* _  _  _  _  _  _  _ *
* _| _| _| _| _| _| _|*
* _| _| _| _| _| _| _|*
*                     *
***********************
***********************
*    _  _  _     _  _ *
*  || ||_||_ |_| _|| |*
*  ||_||_||_|  ||_ |_|*
*                     *
***********************
*****************************************
*                                       *
*  |  |  |  |  |  |  |  |  |  |  |  |  |*
*  |  |  |  |  |  |  |  |  |  |  |  |  |*
*                                       *
*****************************************
*****************
* _  _  _  _  _ *
*|_||_||_||_||_|*
*|_||_||_||_||_|*
*               *
*****************
*****************************************
*    _  _  _  _  _  _  _  _  _  _  _  _ *
*|_|  ||_||_| _|  ||_ |_|  | _||_ |_|  |*
*  |  |  |  | _|  | _|  |  | _| _|  |  |*
*                                       *
*****************************************
********************************************************
* _  _  _  _     _  _  _     _  _  _  _  _     _  _  _ *
* _| _| _| _||_||_||_| _||_||_||_|  | _||_||_||_||_| _|*
*|_  _||_  _|  |  ||_| _|  ||_|  |  | _|  |  ||_|  | _|*
*                                                      *
********************************************************
***********************************************************
* _  _  _     _  _  _  _  _  _  _  _  _  _     _     _  _ *
*|_||_ |_||_|  | _|  | _||_||_ |_||_||_ |_||_||_||_||_| _|  |*
*  | _||_|  |  | _|  ||_ |_| _||_|  ||_|  |  ||_|  | _|  |*
*                                                         *
***********************************************************
```

Problem #9
60 Points

9. Paola

Program Name: Paola.java **Input File: paola.dat**

Test Input File:

```
8
A 3
Z 5
D 7
M 1
J 10
Y 3
R 5
T 8
```

Test Output to Screen:

```
  A
 BC
DEF
    Z
   AB
  CDE
 FGHI
JKLMN
      D
     EF
    GHI
   JKLM
  NOPQR
 STUVWX
YZABCDE
M
         J
        KL
       MNO
      PQRS
     TUVWX
    YZABCD
   EFGHIJK
  LMNOPQRS
 TUVWXYZAB
CDEFGHIJKL
  Y
 ZA
BCD
    R
   ST
  UVW
 XYZA
BCDEF
       T
      UV
     WXY
    ZABC
   DEFGH
  IJKLMN
 OPQRSTU
VWXYZABC
```

Problem #10
60 Points

10. Ricardo

Program Name: Ricardo.java **Input File: ricardo.dat**

Test Input File: (Continuation lines are indented below the first line)

```
30
(rows|curls)
(running,deadlift,stretch)
(lunge,(rows|curls),(squats|press))
(a|a)
((a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(
   a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b),
   (a|b),(a|b),(a|b),(a|b),(a|b),(a|b),(a|b))
((a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c)
   ,(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b
   |c),(a|b|c),(a|b|c),(a|b|c),(a|b|c),(a|b))
((a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d)
   ,(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b|c
   |d),(a|b|c|d),(a|b|c|d),(a|b|c|d),(a|b))
((a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b|c|
   d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),
   (a|b|c|d|e),(a|b|c|d|e),(a|b|c|d|e),(a|b))
((a|b|c|d|e|f),(a|b|c|d|e|f),(a|b|c|d|e|f),(a|b|c|d|e|f),(a|b|c|d|e|f),(a|b|c|d|
   e|f),(a|b|c|d|e|f),(a|b|c|d|e|f),(a|b|c|d|e|f),(a|b|c|d|e|f),(a|b|c|d|e|f),(a
   |b|c|d|e|f),(a|b|c|d|e|f),(a|b|c|d|e|f))
(a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a
   |a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
   a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a)
(a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a
   ,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
   a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a)
a
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
   aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
(((((((((((((((((((((((((((((((((((((((((((((((a|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|
   b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)
   |b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)|b)
(((((((((((((((((((((((((((((((((((((((((((((((a,b),b),b),b),b),b),b),b),b),b),
   b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b),b)
   ,b),b),b),b),b),b),b),b),b),b),b),b)
(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a
   |(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|(a|b)))))))))))
   ))))))))))))))))))))))))))))))))))))
(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a
   ,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,(a,b)))))))))))
   ))))))))))))))))))))))))))))))))))))
```

```
((a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a)
   ,(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a|a),(a|a|a
   |a),(a|a|a|a),(a|a|a|a),(a|a|a),(a|a|a),a)
(mznwqyterupnexvpxzzlvkrkasf,((((((k,y,r,f,ss)|s|s|gk)|gv),((g,kegcup)|lp|c|yvl)
   ),((afh,(v|k),y,rhpc)|(th|zupjn|u|pg)),lq),((m,qnz,q)|v|h),(gujre|i|x|g|ss)))
(npcmljulrtrlhbrbvpql,(((zsnup,dv),sxwxpimxahpfpesiiy),(((lp|(d,cn))|(g,de)|(x|q
   m|y|hz)),a,((v,i),h,d,dx)),(tkr|((t,y),(c|i|o|zi),t,(y|s))),ouddse,xtbd))
(((ot|c)|(((g,wb),((n,xp)|sa))|(g,r,i)|iwbjhci|rz|(o,d,u,p,z))|(oksn,i,c,u,(c|sr
   ))),(xapalyqfuvxlqna|vmmunnguzgnrcb|n|((r|(z|k|dke))|l|b|h|(c|mo))),(xhdm|m|a
   rqx),(j|tho),(wz,n,gt))
(((e|(i,pd)|(e|(c,l,qk)|b|m|z)|tt),y,e,wjy)|((ujl|(z|jp)|(bhis|u|t|t|y)|(yg,m,hx
   )),u,v,mj)|((ttq|((y,d,k,g,m),n,n,eo))|(z|b)|yl|((pb|v|u),(h,c),o,(e|di)))|(c
   |h|cp)|w)
((((tkbzxgtdbjmrtvx,(t,z,ml),((((g|m|i|v|t)|(t,cc))|(b|xq|i|z))|liydpgl|(i|(w|y|
   d)|k))),s),(ixus|a|g|n|ptvv))|((o|(v,d,bt))|d|vi)|((jx,(n,sr),u,(s|x)),n,sz,(
   u,rtv)))
(((((ul,b,l,ix,(l,l)),dpdf,(s|rr))|((i|i)|((r|l|i)|(ut,gry))|(a|t)|przz)),w,e,h)
   |(((zzi,(e|y|ic),h,n)|q|k|k)|s|r|p|rxal)|(((a|v|gt),d,w,(uv|h|gz))|(vtji,mh,e
   x,hymhp)|x|(w|q)))
(((ucqlpfhepgq|w|z|j|hhh),((k,(t|h|z|j)),frin,(ul|v|w))),(((t,g,gg),h,bf,i,(b,ey
   ))|(l|skbn)),(sulpjkgfwoypm,i,f,(v|uf),(w|fv)),(trm,fej),(qv,el,(h,p,n,p),(z|
   b|f)))
(uvdjasludooidmqhhmktwpqxqmaeolmzcqqjew|((d,vm),((gv,(l|y|h|zni))|v|m|t|i),(((x,
   oual,(farf|(n,km)),jwbi,(stq|s|f))|p)|(m|s|p)))|((az,be),(u,(w|y|v|u|ab),(i,t
   )))|((su,lljb),z,r,ep)|osk)
(((zm|(((kr|x|tzv)|(y,xf,q)),(y,c,w),(y|pg))|a|l|rs),(pfxkdd|olc),(s|b|ds))|((s,
   gl),jpvnieuiszakxsdhmqfrcozqinepqgwcwfdvlekaovrcytjmwivykxbyaoozwglrete))
((c|(g,ef,((t,zw)|f|q|g|sj),(k,(g,e),(y,efnf)))|gw|i|nijbbcp)|((z|hj|k|ui)|t|s|g
   s|(k,(h|qey)))|(d|aw)|(tf,(j,q))|((xumw|t|zc)|((i,v,p)|(ctpp|ks)|(z|x))|w))
((x,th),(v|t|mh|(c,fv)),((jo|(t,r,p)|bb|(v,(y,ih))|w),(y,g),(g|d),(nr|e)),mdytsa
   thjdfgmdfagilzxruwtzjxzlfcnwadziqvxqogwebuaqreytznbnthezohorjo)
ytwhqttyqxajgidzevofdkkffumgegxijqhbuswjywvhyoiojcwupksyunitowizmntymqwzwysnuvfn
   dunayfmkvcgrhtpmmwdhiqkaryapmfkowvsfmp
```

~Test Output to Screen on next page~

~Ricardo Output~

Test Output to Screen:

```
Case #1: 2
Case #2: 1
Case #3: 4
Case #4: 2
Case #5: 8589934592
Case #6: 564859072962
Case #7: 549755813888
Case #8: 305175781250
Case #9: 78364164096
Case #10: 99
Case #11: 1
Case #12: 1
Case #13: 1
Case #14: 49
Case #15: 1
Case #16: 49
Case #17: 1
Case #18: 618475290624
Case #19: 1800
Case #20: 63
Case #21: 720
Case #22: 32
Case #23: 86
Case #24: 34
Case #25: 2160
Case #26: 88
Case #27: 73
Case #28: 30
Case #29: 80
Case #30: 1
```

Problem #11
60 Points

11. Shivam

Program Name: Shivam.java **Input File: shivam.dat**

Test Input File:
```
12
f(x)=-x^2
f(x)=x^2
f(x)=-23x^2-25x
f(x)=-56x^2+8x
f(x)=x^2-x
f(x)=x^2+x
f(x)=x^2-1
f(x)=x^2+1
f(x)=-x^2-1
f(x)=-x^2+1
f(x)=x^2-54x+34
f(x)=45x^2-89x+2
```

Test Output to Screen:
```
Function 1: There is one real root at (0.00,0.00).
Function 2: There is one real root at (0.00,0.00).
Function 3: There are two real roots at (-1.09,0.00) and (0.00,0.00).
Function 4: There are two real roots at (0.00,0.00) and (0.14,0.00).
Function 5: There are two real roots at (0.00,0.00) and (1.00,0.00).
Function 6: There are two real roots at (-1.00,0.00) and (0.00,0.00).
Function 7: There are two real roots at (-1.00,0.00) and (1.00,0.00).
Function 8: There are no real roots to the function.
Function 9: There are no real roots to the function.
Function 10: There are two real roots at (-1.00,0.00) and (1.00,0.00).
Function 11: There are two real roots at (0.64,0.00) and (53.36,0.00).
Function 12: There are two real roots at (0.02,0.00) and (1.96,0.00).
```

Problem #12
60 Points

12. Tomek

Program Name: Tomek.java **Input File: tomek.dat**

Test Input File:

```
20
5 5
WW...
WWW..
W.WW.
W.WW.
WWWWW
3 3
...
.W.
...
4 6
.WWWW.
W..W.W
WW.WWW
.WWWW.
4 6
.WWWW.
W.W..W
WWW.WW
.WWWW.
3 3
WWW
WWW
WWW
3 3
...
...
...
4 4
WW.W
.WW.
WWWW
W..W
4 4
WWWW
W..W
W.WW
WWWW
51 50
..................................................
.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.
.W..............................................W.
.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.
.W.W..........................................W.W.
.W.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.W.
.W.W.W......................................W.W.W.
.W.W.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.W.W.
.W.W.W.W..................................W.W.W.W.
.W.W.W.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.W.W.W.
.W.W.W.W.W..............................W.W.W.W.W.
.W.W.W.W.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.W.W.W.W.
.W.W.W.W.W.W..........................W.W.W.W.W.W.
.W.W.W.W.W.W.WWWWWWWWWWWWWWWWWWWWWWWW.W.W.W.W.W.W.
.W.W.W.W.W.W.W......................W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.WWWWWWWWWWWWWWWWWWWW.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W..................W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.WWWWWWWWWWWWWWWW.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W..............W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.WWWWWWWWWWWW.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W..........W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W.WWWWWWWW.W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W.W......W.W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W.W.WWWW.W.W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W.W.W..W.W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
.W.W.W.W.W.W.W.W.W.W.W.W..W.W.W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W.W.WWWW.W.W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W.W......W.W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W.WWWWWWWW.W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.W..........W.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W.WWWWWWWWWWWW.W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.W..............W.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W.WWWWWWWWWWWWWWWW.W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.W..................W.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W.WWWWWWWWWWWWWWWWWWWW.W.W.W.W.W.W.W.
.W.W.W.W.W.W.W......................W.W.W.W.W.W.W.
.W.W.W.W.W.W.WWWWWWWWWWWWWWWWWWWWWWWW.W.W.W.W.W.W.
.W.W.W.W.W.W..........................W.W.W.W.W.W.
.W.W.W.W.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.W.W.W.W.
.W.W.W.W.W..............................W.W.W.W.W.
.W.W.W.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.W.W.W.
.W.W.W.W..................................W.W.W.W.
.W.W.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.W.W.
.W.W.W......................................W.W.W.
.W.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.W.
.W.W..........................................W.W.
.W.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.W.
.W..............................................W.
.WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.
..................................................
20 22
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
W.W.W.W.W.W.W.W.W.W.W.
WWWWWWWWWWWWWWWWWWWWWW
31 41
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.WW.
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
59 26
WWWWWWWWWWWWWWWWWWWWWWWWWW
W........................W
W........................W
```

```
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
W........................W
WWWWWWWWWWWWWWWWWWWWWWWWWW
20 18
W.WW.WWWWWWW.WWWW.
WWWWWWWWWWW.WWWWWW
..W.WWWWWWW.WWW.W.
WWWWWWWWW.WWW.W...
WWWWWWWWWWWWWWWWWW
W.WWWWW.WWW.WWWW.W
WWWWW.WWWWWW.WW.WW
WW.W.WWWWWWWW.WWWW
WWWWW.WWWWWWWWW.WW
WWW...WWWWWWWWWWWW
WWWWW.WWWW.WWWWWWW
WWWW.WWW.WWW.WWWWW
WWWWW.W.WWWW.WWWWW
W.W.WWWWWWWW.W.WWW
WWWWW.WWWW.WWWWW.W
WWWWWWWWWWWWWWWW.W
WW.WWWWWWWW.W.WWWW
WWW..WWW.W.WWWW.W.
.WWW.W..W.WWWWWWWW
...WWW.WWWWWW.WWWW
10 10
.WW.W..WWW
WWW.WWW..W
WWW.WW.WWW
WWW..WWW.W
.WW.WW..W.
W.WWW.W.WW
W.W.WWWWWW
WWW.WWWWWW
.W.WWW.WWW
.WWW.WWW..
10 10
.WWWWWWW.W
WWWWWWWWWW
W.WWW.WW.W
WWWWWWWWWW
W.WWWWWWWW
WWWWWWWWWW
WWWWWWWWWW
.WWWWWWWWW
W..WWW.WW.
WWWW.WWWWW
20 20
WWWWWWW.W.WWWWWWWW.W
WWW.WWWWWWWWWWWWWWWW
WWWWWWWWWW.WW.WWWWWW
WWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWW.WWWW.WW
WWWW.WWWWWWWWW.W.WWW
.WWWWW.WWWWWW.WWWWW.
WWWWWWWWWWWWWWWWWWW.
WW.WWW..WWWWWWWWWWWW
W.WWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWW
WWWWW.WWWWWWWW.WWW.W
.WWW.WWWWWWWWWWW.WWW
WWWWW.WWW.WWW.WWWWWW
WWWWW.WWWWW.WWWWWWWW
WWWWWWWWWWWW.WWWWWWW
.WWWW.WW.WWWWW.WWWWW
WWWWWWWWWWW.WWWWWWWW
WWWWW.WWWWWWWWWWWWW.
WWWWWW.WWWWWWWWWWWWW
20 20
WWWWWWW.WWW.WWW.WWW.
WWWWWWWWW....WWWWWWW
WWW..W..WWW..WWW..WW
WWWW..WWW.WWWWWWW.WW
WWWWWWWWW.WWWWWW..WW
W.WWWWWWWWW.WWWWWWWW
WWWW..W.WWWWW.WWW.WW
.WW.WW..WWWWWW.WWW.W
WWWWWWW.WWWW.WWWW...
WW.WWWWW.WWWW.WWWWWW
WWWWWWWWWWW.WWWWWWWW
WWWWWWWWWW.WWW..W.WW
W.WWWWWWWWWWWW.WWWWW
W.WWWWWWWW.WWW.WW.WW
WWWWWWWWW..WW..WWWW.
.WWWWW.W..WWWW..WWW.
WWWWWWWWWWWWWWW.WWWW
WWW.WW.W.WW..W.WWWWW
WWWWWWWWW.W.WWWWWWWW
WWW.WWWWWWWW..WWWWWW
20 20
WWWWWW.W..WWW.WWWWWW
WWWWWW.WWWW.W.WWW.W.
.WWW.WW.WWWWWWW.WWW.
WW.W..W.WWWWW.WWWWWW
WWWWW.WW..WWWWW.WW.W
.WWWWWWWW.W..WWWWWWW
WWWWW..WWW...WWWWWWW
WWWWWWWW..WWWWWWW..W
W.WW.W.WWW.WW...WWWW
W.WWW.WWWWWWW.WWW.W.
WW.WWWW.W..WW.WWW.W.
.W.WWWWW.WW.W.WW.WWW
.WWWWWWWWWWWWWWWWWWW
WWWWWWW.WWW.WWW.W.WW
.WWWWWWW..W..WWWWW..
W.W.WWW.WW.WWWWWW.WW
WWWWWWWW.WW.WW..W.WW
.WW...WW..WW.WWWWWWW
W.WW.WWWWWW.WWWWWW.W
WWW..WWW.WW..W.WWWWW
```

```
70 100
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWW....................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
W...................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWW..........................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWW..........................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWW.............................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWW..........................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWW..........................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWW..........................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWW..........................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWW..........................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWW..........................WWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWW....................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWW....................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWW....................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWW....................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWW....................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWW.........................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
100 100
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW..............................WWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW..............................WWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW..............................WWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW...............................................................WWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW...............................................................WWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW...............................................................WWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW...............................................................WWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW.................................................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WWWWWWWWWWWWWWWWWWWW..............................WWWWWWWWWWWWWW................................WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW.......................................................................................W......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW......WWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WW........................................................WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW........WWWW
WWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW........WWWW
```

```
WWWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW........WWWW
WWWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW........WWWW
WWWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW........WWWW
WWWWWWWWWWWW.............WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
```

Test Output to Screen: (Continuation lines are indented below first line.)

```
Case #1: 1
2
Case #2: 0
NONE
Case #3: 2
3 1
Case #4: 2
3 1
Case #5: 0
NONE
Case #6: 0
NONE
Case #7: 0
NONE
Case #8: 1
3
Case #9: 24
90 90 82 82 74 74 66 66 58 58 50 50 42 42 34 34 26 26 18 18 10 10 2 2
Case #10: 90
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1
Case #11: 130
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1 1
Case #12: 1
1368
Case #13: 36
5 3 3 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Case #14: 9
3 2 2 2 1 1 1 1 1
Case #15: 6
2 1 1 1 1 1
Case #16: 28
2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Case #17: 30
9 5 5 4 4 3 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Case #18: 43
6 5 4 3 3 3 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1
Case #19: 3
1255 788 361
Case #20: 2
5255 40
```

Computer Science Competition
Region 2022
Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Agustina
Problem 2	Arya
Problem 3	Diego
Problem 4	Fai
Problem 5	Ivan
Problem 6	Juana
Problem 7	Krithika
Problem 8	Michal
Problem 9	Paola
Problem 10	Ricardo
Problem 11	Shivam
Problem 12	Tomek

1. Agustina

Program Name: Agustina.java **Input File: None**

Agustina and her UIL Computer Science Team are working hard this year preparing for the Regional Meet. They do daily practice tests and frequent after-school programming packets. She wants to be reminded of the six teams that won 2020-2021 UIL Computer Science Team State Championships in the six different conferences. This program will print the names of those schools.

Input: None

Output: The names of the six high schools that won the State Championship in each of the six conferences in 2020-2021. Format exactly as shown below.

Sample input: None

Sample output:
```
1A-Slidell HS
2A-San Augustine HS
3A-Fairfield HS
4A-Dallas TAG
5A-Lucas Lovejoy HS
6A-Cypress Woods HS
```

2. Arya

Program Name: Arya.java **Input File: arya.dat**

Arya is very interested in the binary search process. He wants to write a program that will allow him to see the various "guesses" the computer makes when guessing a number using the binary search.

The program will first allow for the input of N representing the largest positive integer in the possible set of numbers 1…N. Then an integer R will be input representing the target number. The program will use a binary search strategy to "find" that value.

Each line of output will display not only each of the program's guesses, but the low value and high values that were used to determine the guesses. All guesses are displayed until the target is reached. For example, if 11-21-31 is displayed, the 21 indicates the guess while the 11 and the 41 indicate the current lower and upper bounds.

If there is an odd number of items in the remaining range of numbers, the guess will be the middle number. For example, if there were 11 numbers remaining, the guess would be the 6th number in that list.

If there are an even number of items remaining, the guess will be the lesser of the two middle numbers. For example, if 20 numbers remain, the guess would be the 10th number in the list.

Input:
The first line of input will contain a single integer T, the number lines of data to follow (1 <= T <= 10).
Each line of data will consist of two positive integers N and R. N (1<=N<=100000) will represent the largest possible integer in the range. R represents the target and will be in the range (1<=R<=N).

Output:
For each test case, sets of 3 integers will be printed on separate lines until the target is found. The first is the low value of the current range. The second is the computer's guess. The third is the high number in the current range. A dash (-) will separate the numbers with no extra spaces. After the target is found, "GOT IT!!!" will be printed.

Sample input:
```
5
10 7
15 12
100 32
1000 500
31 31
```

Sample output:
```
1-5-10
6-8-10
6-6-7
7-7-7
GOT IT!!!
1-8-15
9-12-15
GOT IT!!!
1-50-100
1-25-49
26-37-49
26-31-36
32-34-36
32-32-33
GOT IT!!!
1-500-1000
GOT IT!!!
1-16-31
17-24-31
25-28-31
29-30-31
31-31-31
GOT IT!!!
```

3. Diego

Program Name: Diego.java **Input File: diego.dat**

Diego has found a fancy digital safe but is having trouble opening it. The combination for the safe is a series of digits (0-9). Each time he tries a combination, the safe gives him two pieces of information:

1) The number of digits that belong in the correct combination from those that he picked.
2) The number of digits that are in the correct position from those that he picked.

Note that excess digits are not counted in total (1). For instance, if the code was `123`, and Diego guessed `411`, the safe's response would be `1 0` rather than `2 0`. Given a set of Diego's guesses and the safe's responses, write a program to help Diego determine how many combinations could possibly open the safe.

Input:
The first line of input will contain a single integer T, the number of test cases to follow (1 <= T <= 10)
The first line of each test case will contain two space separated integers N and M denoting the length of the safe's code (1 <= N <= 6) and the number of observations (1 <= M <= 10)
The next M lines of each test case will each consist of a single observation of the form G X Y, where G is a guess consisting of N digits (0-9), and X and Y are integers denoting how many digits belong in the code, and how many digits are in the correct position.

Output:
For each test case on its own line, output the number of safe combinations that are consistent with Diego's observations.

Sample input:
```
2
2 4
02 1 0
34 0 0
56 0 0
78 0 0
5 2
12345 0 0
67890 0 0
```

Sample output:
```
4
0
```

Sample Explanation:
Combinations that are consistent with test case 1: 10, 90, 21, 29

There are no combinations consistent with test case 2, because every digit was used and the safe reported that none of them are present in the combination

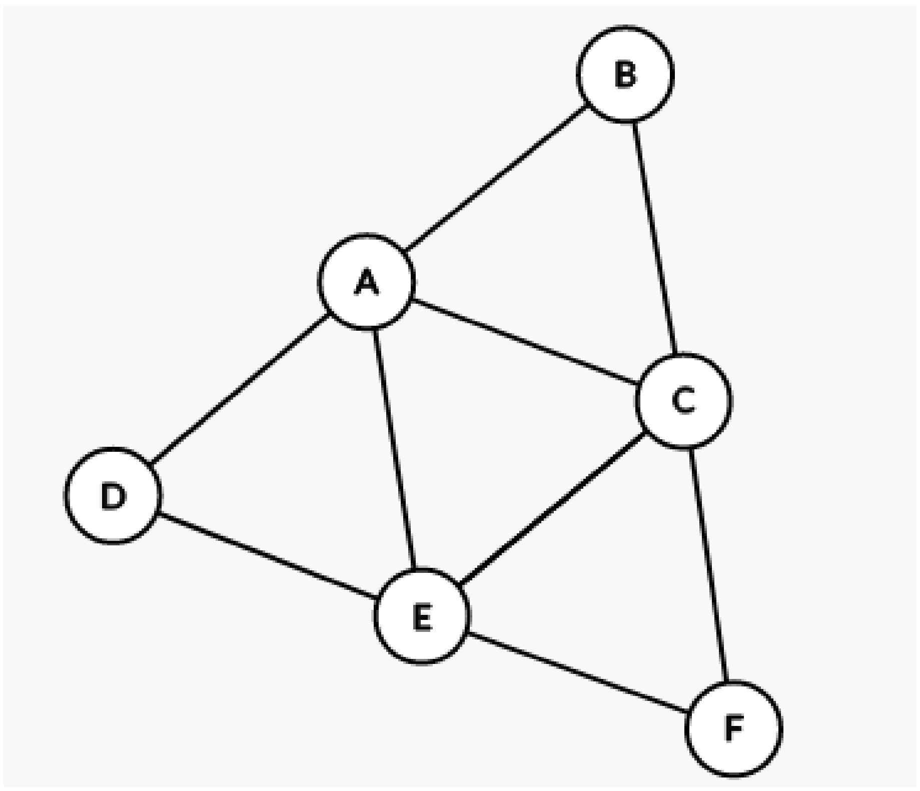
4. Fai

Program Name: Fai.java **Input File: fai.dat**

Fai just got a job working for the UIL's virtual reality team. In an attempt to get all K-12 students more familiar with their assigned district, the UIL wants to make a warped speed video showing what each student would see on a bus ride to another school's city or town. This way, students will have a general idea as to what they will see out of their school bus window before ever making the trek to the other campus.

To accomplish this, the UIL virtual reality team has provided Fai with a 360-degree camera attached to the top of her car. Fai knows she needs to drive every single road between two cities exactly one time to record the street view, so that all possible routes have been recorded. With the rising price of fuel, and in an attempt to be as efficient as Fai can be with her time, Fai knows she needs to start recording and end recording in the same city, if possible. With the task of driving every single road, Fai knows she may visit the same city or town multiple times, which is okay, she just has to travel all the given roads the UIL assigns to her to drive.

For example, say that City A, City B, City, C, City D, City E, and City F are all in the same district and the following roads exist: City A<->City B,City A<->City C,City A<->City E,City A<->City D,City B<->City C,
City C<->City F,City C<->City E,City F<->City E,City E<->City D. If Fai were to start in City A she could do the following road traversal to cover all roads exactly once, starting and stopping from City A:



City A -> City D -> City E -> City F -> City C -> City B -> City A -> City E -> City C -> City A

Can you help Fai write a program that, given a district and all the roads between cities or towns, can determine if it's possible to drive all roads starting and stopping from a given city or town?

Continued next page…

Fai, continued

Input: The input will consist of an integer *T*, the number of test cases. *T* will be in the range of [1,10]. For each test case, input will consist of four lines. Line 1 will contain the name of all the cities or towns in the district. The number of cities or towns will be greater than or equal to two, and will not exceed 10.. Cities or towns are not limited to one word names. It will be guaranteed that no two cities or towns have the same exact name. Names in the list will be separated by a comma ",". Line 2 contains the name of the start city or town where Fai will begin recording street views. Line 3 will consist of the roads Fai has been assigned to record by the UIL. The roads will be given in the form: "`Name1<->Name2`" this means a road exists between Name1 and Name2 and must be driven by Fai exactly once. A road in this problem is always two-way. Roads will be separated by a comma ",". There will be at least one road present and there will always exist a path, or series of road(s), between two cities or towns. Line 4 is 20 dashes and serves to separate all test cases.

Output: For each test case, you are to output "Test Case #: possible" if Fai can start and stop at the given start city, traversing each road exactly once or "Test Case#: impossible" if Fai cannot start and stop at the given start city, traversing each road exactly once.

Sample input:
```
9
City A,City B,City C,City D,City E,City F
City A
City A<->City B,City A<->City C,City A<->City E,City A<->City D,City B<->City C,City C<->City F,City
C<->City E,City F<->City E,City E<->City D
--------------------
Town 0,Town 1,Town 2,Town 3,Town 4
Town 3
Town 1<->Town 2,Town 1<->Town 0,Town 2<->Town 0,Town 0<->Town 4,Town 0<->Town 3,Town 3<->Town 4
--------------------
Dallas,San Antonio,Houston,El Paso,Austin
Houston
Dallas<->San Antonio,Dallas<->Houston,Dallas<->El Paso,Dallas<->Austin,San Antonio<->Houston,San
Antonio<->El Paso,San Antonio<->Austin,El Paso<->Austin
--------------------
Wall,College Station,San Angelo
Wall
Wall<->College Station,College Station<->San Angelo,San Angelo<->Wall
--------------------
Lubbock,Loredo,Lampasas,Liberty Hill,Lago Vista
Lubbock
Lubbock<->Loredo,Lubbock<->Lampasas,Lubbock<->Liberty Hill,Lubbock<->Lago Vista
--------------------
Abilene,Beaumont,Childress,Dalhart,Eden,Fort Worth
Beaumont
Abilene<->Fort Worth,Abilene<->Eden,Abilene<->Childress,Abilene<->Beaumont,Beaumont<-
>Childress,Childress<->Dalhart,Childress<->Eden,Dalhart<->Eden,Eden<->Fort Worth
--------------------
Alice,Big Spring,Colorado City,Denton,Eagle Pass,Frenship,Goliad
Frenship
Alice<->Big Spring,Alice<->Colorado City,Big Spring<->Colorado City,Big Spring<->Denton,Big Spring<-
>Eagle Pass,Colorado City<->Denton,Colorado City<->Frenship,Denton<->Eagle Pass,Denton<-
>Frenship,Frenship<->Eagle Pass,Frenship<->Goliad,Eagle Pass<->Goliad
--------------------
Caldwell,Bryan,Navasota,Hempstead,Franklin,Calvert
Bryan
Caldwell<->Bryan,Bryan<->Navasota,Navasota<->Hempstead,Hempstead<->Franklin,Franklin<-
>Calvert,Calvert<->Hempstead,Bryan<->Calvert
--------------------
Texline,Brownsville
Texline
Texline<->Brownsville
--------------------
```

Sample output:
```
Test case 1: possible
Test case 2: possible
Test case 3: impossible
Test case 4: possible
Test case 5: impossible
Test case 6: possible
Test case 7: possible
Test case 8: impossible
Test case 9: impossible
```

5. Ivan

Program Name: Ivan.java **Input File: ivan.dat**

Ivan has worked on an algorithm for scoring UIL written exams. All questions will be multiple choice with 5 choices, only one that is correct. Correct answers will earn 6 points while incorrect answers will be penalized 2 points. Questions that are unanswered earn no points. He is unsure of his string handling skills and needs you to validate his algorithm.

Can you help Ivan implement his scoring algorithm?

Input: First line of data file contains a positive integer T, the number of exams that will be scored with $1 \le T \le 25$. The next line will contain a single string with exactly 40 uppercase letters from { A, B, C, D, E }. Each letter is the correct answer for one of the 40 questions with the first letter for the first question and the remaining letters in sequence to the last question. The following T lines will then contain a single string with exactly 40 uppercase letters from the same set or an underscore '_' which indicates the question was not answered.

Output: For each exam, display one line with the score and the percentage of attempted questions that were correct. Format the line as shown below with the percentage correct rounded to 1 decimal place. If no questions are attempted, set the percentage to 0.0.

Sample input:
```
4
BBEEBCECDCCCCDDBDAEBBBECAEBCBBECAEBCDEED
BBEEBCECDCCCCDDBDAEBBBECAEBCBBECAEBCDEED
BCEEBAECDCDCCDDADAEBEB_CAEBCB_ECAE_CD___
AB_DEA_CDEAB_DEABCD_ABCDEAB_DEABC_EABCD_
BACBEDACBACBEDAAECBDBADCDBBAACCDABECBAAD
```

Sample output:
```
Exam #1: 240 100.0
Exam #2: 164 85.3
Exam #3: -18 18.2
Exam #4: 0 25.0
```

6. Juana

Program Name: Juana.java **Input File: juana.dat**

Juana has been working with 2-dimension tables of data and would like to extract an arbitrary "chunk" of the data. She will provide the number of rows and columns in the original table along with the data to populate the table. She will then provide a starting point by identifying the row and column positions of the top left corner of the "chunk" along with the numbers of columns and rows desired. Juana does not have programming experience so her tables start with row 1 and column 1 at the top-left corner. She has provided the following example:

	Col 1	Col 2	Col 3	Col 4	Col 5
Row 1	1	2	3	4	5
Row 2	6	7	8	9	10
Row 3	11	12	13	14	15
Row 4	16	17	18	19	20

The table contains 4 rows and 5 columns with the data as shown above. Juana would like to extract the "chunk" of data that starts at row 3 and column 2. She wants 3 columns across the rows and 2 total rows as shown by the shading above.

Can you create a "chunk" extraction program for Juana?

Input: First line of data file contains a positive integer T, the number of test cases that follow with $1 \leq T \leq 10$. Each test case starts with a line containing the number of rows R and columns C for the table with $2 \leq R,C \leq 15$. That line will then be followed by R lines of data each containing C integers in the range [0, 1000). The data items will be right-aligned with leading spaces for student viewing below. The next line will contain 4 integers separated by a space. The first pair of integers are the row and column numbers of the top-left corner. The next 2 integers are the count of columns and rows desired in the "chunk". All 4 integers will have the same range as specified for R and C above. There is no guarantee that Juana specifies the starting point and sizes for the "chunk" correctly. When there is not enough data to extract the complete "chunk" an error message will be displayed instead of the requested "chunk".

Output: For each test case, display a line with the test case number formatted as shown in the sample. If the "chunk" cannot be extracted, display the error message as shown below on the same line as the test case. Otherwise, the extracted "chunk" of data is displayed below the test case line. Each data item is right-aligned in a column that is 5 positions wide. Follow the "chunk" with a line containing 20 hyphens "--------------------".

~ Sample input and output on next page ~

Juana, continued

Sample input:

```
3
4 5
    1     2     3     4     5
    6     7     8     9    10
   11    12    13    14    15
   16    17    18    19    20
3 2 3 2
7 6
    1     2     3     4     5     6
    7     8     9    10    11    12
   13    14    15    16    17    18
   19    20    21    22    23    24
   25    26    27    28    29    30
   31    32    33    34    35    36
   37    38    39    40    41    42
5 4 2 3
3 3
    1     2     3
    4     5     6
    7     8     8
2 2 3 1
```

Sample output:

```
Test case #1:
   12   13   14
   17   18   19
--------------------
Test case #2:
   28   29
   34   35
   40   41
--------------------
Test case #3: Unable to extract requested size!
--------------------
```

7. Krithika

Program Name: Krithika.java **Input File: krithika.dat**

You are given an array A of length N. An integer X is a k-"array factor" of A if the bitwise AND of some k elements (not necessarily consecutive) of A is equal to X. Given A and k, find the largest k-"array factor" of A.

Input:
The first line of input is T (1 <= T <= 50), the number of test cases. The first line of each test case has space-separated integers N and k, where N (1 <= N <= 1,000) is the number of elements in the array and k (1 <= k <= N) is the sought k-array factor. The second line of each test case contains N positive integers in the range [1, 10^18], the elements of A. Note that the elements of A may not fit into a 32-bit integer data type.

Output:
For each test case, output the largest k-"array factor" of A. Format the output with the case numbers as in the samples.

Sample input:
```
3
3 2
14 159 26
4 3
1 1 1 2
5 4
1 2 3 4 5
```

Sample output:
```
Case #1: 26
Case #2: 1
Case #3: 0
```

Sample Explanation:
The 2-array factors of the first array are 14, 10, and 26. Of these, 26 is the largest.

8. Michal

Program Name: Michal.java **Input File: michal.dat**

Michal's school just got a new scrolling marquee sign, but the problem is, the sign didn't come with a controller! Michal's principal has asked Michal to write a program to convert all input into text to be displayed on the sign. Michal knows this is a big undertaking, so he has decided to start with only the numerical digits 0-9 first.

The sign utilizes a seven-segment display. For example, if all 7 segments are turned on, you would get the numeral 8 as shown below:

```
*****
* _ *
*|_|*
*|_|*
*   *
*****
```

With a combination of turning certain segments on or off, all digits 0-9 can be achieved as the table below shows:

```
*****  *****  *****  *****  *****  *****  *****  *****  *****  *****
* _ *  *   *  * _ *  * _ *  *   *  * _ *  * _ *  * _ *  * _ *  * _ *
*| |*  *  |*  * _|*  * _|*  *|_|*  *|_ *  *|_ *  *  |*  *|_|*  *|_|*
*|_|*  *  |*  *|_ *  * _|*  *  |*  * _|*  *|_|*  *  |*  *|_|*  *  |*
*   *  *   *  *   *  *   *  *   *  *   *  *   *  *   *  *   *  *   *
*****  *****  *****  *****  *****  *****  *****  *****  *****  *****
```

Outputting one number at a time wouldn't be too hard programmatically, a switch statement or if-else statement could get the job done in a jiffy. The problem is, Michal's principal wants to utilize the sign's full potential and display multiple numbers at one time. For example, if Michal's principal wanted to display "0123456789" the sign would need to display:

```
********************************
* _     _  _     _  _  _  _  _ *
*| |  | _| _||_||_ |_   ||_||_|*
*|_|  ||_  _|  | _||_|  ||_|  |*
*                              *
********************************
```

Given an input number, can you help Michal write a program that outputs the input number on a single line surrounded by a box of asterisks ('*')?

Input: The input will consist of an integer *T*, the number of test cases. *T* will be in the range of *[1,20]*. For each test case, input will consist of a single number with a minimum of one digit, and a maximum of 20 digits. The input will have no spaces and will consist of the digits [0-9] only.

Output: For each test case, you are to output the number using the seven segment table above, in a single bounded box of asterisks

~ Sample input and output on next page ~

Michal, continued

Sample input:

```
7
8
0123456789
9876543210
2022
01010001
246810
13579
```

Sample output:

```
*****
* _ *
*|_|*
*|_|*
*   *
*****
********************************
* _     _  _     _  _  _  _  _ *
*| |  | _| _||_||_ |_   ||_||_|*
*|_|  ||_  _|  | _||_|  ||_|  |*
*                              *
********************************
********************************
* _  _  _  _  _     _  _     _ *
*|_||_|  ||_ |_ |_| _| _|  || |*
*  ||_|  ||_| _|  | _||_   ||_|*
*                              *
********************************
**************
* _  _  _  _ *
* _|| | _| _|*
*|_ |_||_ |_ *
*            *
**************
**************************
* _     _     _  _  _     *
*| |  || |  || || || |  |*
*|_|  ||_|  ||_||_||_|  |*
*                        *
**************************
********************
* _     _  _     _ *
* _||_||_ |_|  || |*
*|_   ||_||_|  ||_|*
*                  *
********************
*****************
*    _  _  _  _ *
*  | _||_   ||_|*
*  | _| _|  |  |*
*               *
*****************
```

9. Paola

Program Name: Paola.java **Input File: paola.dat**

Paola absolutely loves isosceles right triangles. She also is intrigued by the alphabet. This program will combine those two things. You will create a program that creates a right triangle using a continuous string of letters. The program will accept the input of an uppercase letter representing the starting letter. It will also accept the input of the number of rows to be created. The program will then display a right triangle of the shape shown below - with a hypotenuse having a positive slope.

```
     *
    **
   ***
  ****
 *****
******
```

Instead of asterisks, the program will output letters of the alphabet beginning with the input character. The letters will proceed in order as shown below. If the letter 'Z' is reached, the next letter will be an 'A'. In the example below, the inputs were M and 6.

```
     M
    NO
   PQR
  STUV
 WXYZA
BCDEFG
```

Input:
The first line of input will contain a single integer T, the number data lines to follow (1 <= T <= 10).
Each line of data will consist of an uppercase letter Ch ('A' <= Ch <= 'Z'), and an integer N (1 <= N <= 20) representing the number of rows that the triangle will use.

Output:
For each test a triangle will be produced.

Sample input:

```
5
A 3
Z 5
D 7
M 1
J 10
```

Sample output:

```
  A
 BC
DEF
    Z
   AB
  CDE
 FGHI
JKLMN
      D
     EF
    GHI
   JKLM
  NOPQR
 STUVWX
YZABCDE
M
         J
        KL
       MNO
      PQRS
     TUVWX
    YZABCD
   EFGHIJK
  LMNOPQRS
 TUVWXYZAB
CDEFGHIJKL
```

10. Ricardo

Program Name: Ricardo.java **Input File: ricardo.dat**

Ricardo has been following his New Years resolution of regularly going to the gym. At this point in time, he has gotten comfortable with many different exercises. In fact, he is comfortable with so many exercises, he has decision paralysis and cannot decide which exercises to do when he goes to the gym.

To help with this, Ricardo has developed "workout plans". Each workout plan is one of the following:

1. A single exercise (e.g. "`squats`"). All exercise names are made of lowercase English letters.
2. An option between multiple workout plans. These are surrounded by parentheses and separated by '`|`' characters. (e.g. "`(rows|curls)`"). This means that Ricardo can choose to either do rows or curls.
3. A sequence of workout plans in order. These are surrounded by parentheses and separated by '`,`' characters. (e.g. "`(running,deadlift,stretch)`"). This means that Ricardo runs, then does deadlifts, then stretches.

Since workout plans can nest, these plans can get quite complicated, and there can be many options. Now, Ricardo is wondering, how many different workouts can he complete given a plan? Two plans are different if Ricardo makes a different decision when presented with an option. Note that different exercises can have the same label. For example, there are two different workouts for the plan "`(a|a)`".

Input:
The first line of input is T (1 <= T <= 30), the number of test cases. Each test case is a single workout plan. Each workout plan consists only of lowercase letters, parentheses, and the '`|`' and '`,`' characters. No workout plan has more than 200 characters.

Output:
For each test case, output the number of workout plans that Ricardo can do. Format your answer with the case number as in the samples. It can be proven that given the bounds on the input data, the total number of workouts will fit into a signed 64-bit integer data type.

Sample input:
```
4
(rows|curls)
(running,deadlift,stretch)
(lunge,(rows|curls),(squats|press))
(a|a)
```

Sample output:
```
Case #1: 2
Case #2: 1
Case #3: 4
Case #4: 2
```

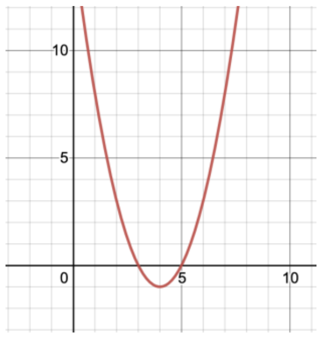
Sample Explanation:
In the third test case, these are the possible workouts:

1. lunge, rows, squats
2. lunge, rows, press
3. lunge, curls, squats
4. lunge, curls, press

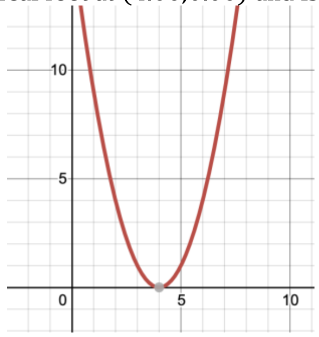
11. Shivam

Program Name: Shivam.java **Input File: shivam.dat**

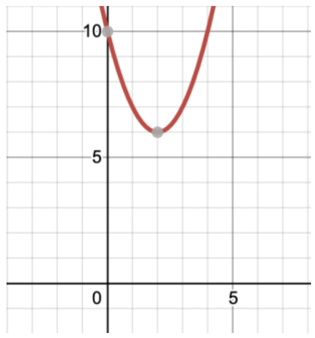
In Shivam's Algebra I class, Shivam's teacher just taught his class about quadratic functions. A quadratic function is any function that can be written in the form: $f(x) = ax^2 + bx + c$ where x represents an unknown variable, the coefficients of the function are a, b, and c, and $a \neq 0$. When graphed on the xy plane, quadratics are known for their "U" shaped appearance. For example, the function $f(x) = x^2 - 8x + 15$ is graphed below:



Where the function intersects the x axis, of the xy plane, is known as the function's root(s). In the above example, the function has two real roots, one at (3.00,0.00) and another at (5.00,0.00). Not all quadratics have two real roots, though. For example, the function $f(x) = x^2 - 8x + 16$ only has one real root at (4.00,0.00) and is graphed below:



Some quadratics have no real roots meaning their graph does not intersect the x axis at all. For example, the function $f(x) = x^2 - 4x + 10$, which is graphed below, shows an example of a quadratic that doesn't intersect the x axis at all.



Shivam needs your help writing a program that can read in a quadratic function $f(x)$, determine the number of roots, and where those roots are. Can you help him with this?

Continued next page…

Shivam, continued

Input: The input will consist of an integer *F*, the number of functions. *F* will be in the range of *[1,20]*. The following *F* lines will each contain a single function $f(x)$ of the form `f(x)=ax^2+bx+c.` There will be no spaces in the function input. For this program, the caret operator (`^`) will be used for exponents and not the `xor` operator. `a` will be in range of $[-100,0) \cup (0,100]$, `b` and `c` will be in range $[-100,100]$. a, b, and c are all guaranteed to be integers, but b and/or c are not guaranteed to be present in the function input. For example, the function `f(x)=4x^2+8` is a legal input in which only coefficients a and c are present.

Output: For functions with two real roots, you are to output "`Function #: There are two real roots at (ROOT1_X,ROOT1_Y) and (ROOT2_X,ROOT2_Y).`" Roots should be displayed in ascending order according to the x component and rounded to two decimal places. For functions with one real root, you are to output "`Function #: There is one real root at (ROOT1_X,ROOT1_Y).`" The root should be rounded to two decimal places. For functions with no real roots, you are to output "`Function #: There are no real roots to the function.`"

Sample input:
```
9
f(x)=x^2-8x+15
f(x)=x^2-8x+16
f(x)=x^2-4x+10
f(x)=-23x^2-25x
f(x)=4x^2+8
f(x)=-78x^2+32x+6
f(x)=-89x^2+6
f(x)=3x^2+54
f(x)=x^2
```

Sample output:
```
Function 1: There are two real roots at (3.00,0.00) and (5.00,0.00).
Function 2: There is one real root at (4.00,0.00).
Function 3: There are no real roots to the function.
Function 4: There are two real roots at (-1.09,0.00) and (0.00,0.00).
Function 5: There are no real roots to the function.
Function 6: There are two real roots at (-0.14,0.00) and (0.55,0.00).
Function 7: There are two real roots at (-0.26,0.00) and (0.26,0.00).
Function 8: There are no real roots to the function.
Function 9: There is one real root at (0.00,0.00).
```

12. Tomek

Program Name: Tomek.java **Input File: tomek.dat**

The search for new habitable land is on! The Universe's Inhabitation Legion (UIL) is on the hunt for new places for the galaxy's displaced to live. Currently, the UIL is investigating the distant planet Xae-12, which was thought to be inhospitable to humans.

On Xae-12, humans cannot survive on the mainland because of human-hunting predators. However, the UIL has found a large lake. Inside the lake are islands which are devoid of any predators. These islands warrant further investigation, but the UIL only has grainy images to work with. Each image is an R by C grid of tiles, and each tile is either water or land. Two tiles are connected if they share an edge. In the photos the UIL has taken, there is at most one connected body of water (the lake), and potentially many connected bodies of land. A connected region of land is an island if it is fully within the lake. Any land on the border of the image can be assumed to be mainland, and not an island.

Can you write a program to help the UIL find all islands in the lake, and their sizes? The size of an island is the number of cells it takes up in the image. Output the sizes in descending order.

Input:
The first line of input is T (1 <= T <= 20), the number of test cases. Each test case starts with two integers R and C (3 <= R, C <= 100), the number of rows and columns in the input image respectively. Then follow R lines with C characters each. All characters are either `'.'`, signifying land, or `'W'`, signifying water. In each test case, there is at most one connected body of water.

Output:
For each test case, output two lines. On the first line, output the case number and the total number of islands, formatted as in the samples. On the second line, output a space separated list of island sizes. Output the island sizes in descending order. If there are no islands in the image, output the string "`NONE`".

Sample input:
```
3
5 5
WW...
WWW..
W.WW.
W.WW.
WWWWW
3 3
...
.W.
...
4 6
.WWWW.
W..W.W
WW.WWW
.WWWW.
```

Sample output:
```
Case #1: 1
2
Case #2: 0
NONE
Case #3: 2
3 1
```

Sample Explanation:
In the first test case, there are two masses of land. The landmass in the upper right is part of the mainland, so there is only one island of size 2.

In the second test case, there are no islands.

In the third test case, there are two islands. The land in each corner of the image is part of the mainland.