Program 4
Loops, Decision Statements, Symbolic Constants, and Random Number Generation
(100 points)

Due: Friday, September 29 on Blackboard by 11:59 PM

Overview

For this assignment, write a program that will generate three sets of random numbers.

Random Number Generation

In the first three programs, the user was asked for input. This program will be different. Rather than asking the user how many values are in a set of numbers or even what the values are, a random number generator will be used to determine the size of a set and the actual values in the set.

To use the random number generator, first add a #include statement for the cstdlib library to the top of the program:

```
#include <cstdlib>
```

Next, initialize the random number generator. This is done by calling the ***srand*** function and passing it an integer value (known as a ***seed value***). This should only be done ONE time and it must be done BEFORE actually generating a random number. A value of 1 (or any integer literal) will generate the same sequence of "random" numbers every time the program is executed. This can be useful for debugging:

```
srand(1);
```

To get a different series of random numbers each time the program is run, the time that the program is run can be passed as the seed value for the random number generator. This is done as follows:

```
srand(time(0));
```

If the time function is used, make sure to #include the ctime library as well.

Note: the two srand instructions that are listed above are simple examples of how to use the instruction. In a program, ONLY ONE version will be used.

Now that the random number generator has been initialized, a random number can be generated by calling the ***rand*** function:

```
num = rand();
```

The above line of C++ code will generate a "random" integer between 0 and RAND_MAX and saves the value in an integer variable named num. RAND_MAX is a pre-defined constant that is equal to the maximum possible random number. It is implementation dependent but is guaranteed to be at least 32,767.

Modulus division can be used to restrict the "random" integer to a smaller range.

To generate a value between 0 and 11: `num = rand() % 12;`

To change the range to 1 through 12, simply add 1: `num = rand() % 12 + 1;`

To get random values that are within a specified range that starts at a value other than 0 or 1: `num = minimum_value + (rand() % (maximum_value - minimum_value + 1));`

So, to get values within the range 8 - 17: `num = 8 + (rand() % (17 - 8 + 1));`

To convert a random integer value to a random double value: `double_num = minimum_value + (rand() / (RAND_MAX / (maximum_value - minimum_value)));`

where minimum_value and maximum_value are both double values. So, to get values within the range 0.5 - 230.5: `double_num = 0.5 + (rand() / (RAND_MAX / (230.5 - 0.5)));`

Basic Program Logic

Initialize the random number generator using a seed value of 17. Other seed values may be used to produce different results. However, ***the version that is handed in for grading MUST use a seed value of 17***.

The first set of numbers should have exactly 52 values. Display the number of values in the first set with a label.

In a ***for*** loop that executes exactly fifty-two times:

- generate a random number (no restrictions)
- display the random number *

* when displaying the random numbers, make sure that there are exactly 6 values displayed per line. The exception is the last line, which may have less than 6 values displayed.

Next, generate a random number between 1 and 80. This will be the number of values in the second set. Display the number of values in the set with a label.

In a ***while*** loop that executes exactly "number of values in the second set" number of times:

- generate a random number (no restrictions)
- display the random number **

** again, make sure that there are exactly 6 values displayed per line.

Finally, generate a random number between 1 and 100. This will be the number of values in the third set. Display the number of values in the set with a label.

In a ***do while*** loop that executes exactly "number of values in the third set" number of times:

- generate a random double number between 0.0 and 200.0
- display the random number with 4 digits after the decimal point ***

*** again, make sure that there are exactly 6 values displayed per line.

Note: when writing this program, it is important that the steps that involve the random number generator are executed in the sequence that they're listed above. This is because the random number generator simply generates a sequence of values and if those values are not processed in the same order as above, the results will not match the expected results that are listed in the Output section below.

Symbolic Constants

This program MUST use at least 6 symbolic constants.

The first constant represents the size of the first set of values (the one generated by the for loop). It should have an integer value of 52.

The second constant represents the maximum size of the second set of values (the one generated by the while loop). It should have an integer value of 80.

The third constant represents the maximum size of the third set of values (the one generated by the do while loop). It should have an integer value of 100.

The fourth constant represents the minimum random double value. It should have a value of 0 (make sure to use 0.0 if using #define to create the constant).

The fifth constant represents the maximum random double value. It should have a value of 200 (make sure to use 200.0 if using #define to create the constant).

The sixth constant represents the number of values to display on a line of output. It should have an integer value of 6.

More symbolic constants may be added to the code if necessary.

Program Requirements

1. Include line documentation. There is no need to document every single line, but logical "chunks" of code should be preceded by a line or two that describes what the "chunk" of code does. ***This will also be a part of every program that is submitted for the remainder of the semester.***
2. To use the random number generator, add #include <cstdlib> at the beginning of the program
3. The program MUST use the 6 symbolic constants described above. Make sure to follow the programming convention of capitalizing the names of constants.
4. Make sure that the copy of the program that is handed in uses srand(17); to set the seed value for the random number generator.
5. The numbers in each set MUST be displayed in columns with the LAST digit of the values lined up.
6. Hand in a copy of the source code (the CPP file) using Blackboard.

Output

Run 1 (using srand(17);) on Windows PC

```
Set 1 has exactly 52 values
          94        26602        30017        18297        20363        13015
       28509        15290        29003        24399         3339        28849
       17055        19424         4588        15756         6098        11834
        1351        21383        18431          155        14763        14082
        4564        25482        30678        20183        15765        18376
       20694        32234         8292        29828        23406        31490
       25791        15822        24763        23255        15434        32590
       15383          108        24271        12086        29246        26526
       31363         7541        17328        32253

Set 2 has exactly 76 values
       13102        15958        32220        16574        18024         4711
       32676        19373        22023        11917        13678        18912
       16891        26436         1232         8547        21431        11651
       29430        31362        10980         6303         5877        14544
       25175        18284         1629         1390          321        23717
       28685        20105         3238         6798        22860        24613
       19729        25047        12565        31257         7301         6084
       11697         1883        20662        14219        10380        23166
        5497         4862        27713         4985        19322         1641
       10487         4778        16173         9150          752         7451
       19463         6383        28511         5635        16365        21916
       21370        29072         4845        13294         6609        17635
       18220        10435        12766         5981

Set 3 has exactly 12 values
      98.5076      71.6636     139.2193     112.4912     153.8194     179.4855
      54.2558      72.5974      95.6999      73.5862      53.7492     105.6612
```

Run 2 (using srand(17);) on a Mac

```
Set 1 has exactly 52 values
     285719    507111939   1815247477   1711656657    122498987   1551140683
 1717468248   1161144809   1176904574   1910786348   1115693598   1782579629
  271465306   1262131714   1951735779     10529728    879479442    281039393
 1112538398    292740757    208867622   1449843756     27064583   1755396964
  826431462   2056836685   1209899036    250444609    154595343   1976200578
 1021029944   2055929278    991495116   1732797539   1102501006   1245501526
 1621040173   1844641769   1820283491    452598075    428768851   1510443072
  612519917   1733124948    162813128    506076018   1584392406    85944842
 1365948710    899782540     65307606    260790425

Set 2 has exactly 9 values
 1446150460   230864474   1783748036    581528932    558682627    980407305
   63551704   814116569   1238860146

Set 3 has exactly 58 values
     84.7120     154.5832      79.5207     104.8913     108.4918      22.3984
     50.7469     102.4041     106.0833     142.0781     106.9793       1.1411
    179.1158     199.3318     170.2855     189.1485      18.6955      14.5788
     26.2423      53.9502     140.7480     152.0402     140.4476     103.1633
     66.4008     198.1568      21.1455     192.8626      41.4942     193.4518
    145.1306       9.7862      76.3373       1.0746      61.1362     115.6787
     12.3417      26.8335     190.9191     176.5221       6.3142     122.6479
    143.6264     128.1795     112.3828      17.1295      94.8324      48.2266
    144.1950      85.5851      28.5074     124.4876      62.9347     143.2096
    123.3552      30.0139      43.4527     109.3671
```

Run 3 (using srand(17);) on onlinegdb

```
Set 1 has exactly 52 values
   1227918265      3978157    263514239   1969574147   1833982879    488658959
    231688945   1043863911   1421669753   1942003127    134395001    461983965
    602354579    726141576   1746455982   1641023978   1153484208    945487677
   1559964282   1484758023    360805106   1295207561    80815853   1578628148
   1766416483    134664533   1026318808    487190350   2084406199    494281178
    767599596   1164840817    498259336   1031113836    986931316    184758567
   1519772795   1218620261   1228622478    793958901   1013139741    425093831
   1255942866   1615494320   1151235407    854915200   1109034650    157235967
   1800402877    521515284   1641993990     13724335

Set 2 has exactly 46 values
    302665195   1592352484   1433855680    437329729    471187644   1921046030
    374252280    965468822    541161979   1539093097   1463728158   1572275815
    378540765   1648486725    944564962   1597161027    729625555   1738523863
    462817120   1154719386    846983081   2078311440    158471146   1701898282
   1039862443    315707113   1354817511   1561377727   1957701104   1368541847
   1230616925    122882651    813410683    516988957    550212380   1284598327
    290551340    924464661    102583501    831713319    316074110   1566311660
    256505486    694614876   1067314737   1201070448

Set 3 has exactly 56 values
     167.3531      73.7711      56.5415      74.8947     152.6525      50.0993
      89.6535     111.1541     146.9440     119.0560      37.3313      92.3587
     101.3811     164.7867       6.9688     111.8942      40.5415      55.1172
     163.1367     160.1790      82.1769      49.2342     169.7329     159.6362
      78.6709     115.6070     183.5251     143.3619      15.0085      95.3835
     156.8002     182.3616     169.1546      13.3416      57.2563     121.8071
      63.4409     146.9098      32.9612      10.3850      65.9658      70.2926
     102.7437     167.3470      35.0793     109.7125      79.2411      75.6208
     164.8296      42.3778      35.7999      47.0065      91.6120       5.5327
       6.6427     170.2828
```