Questions on 09/14/2023

September 17, 2023

Contents

- Section 4.8, nested loop
- Section 4.9-4.10, random number

Problem 1: Generate a random int number in [200, 800].

Answer:

1. There are a total of (800 - 200 + 1) integers in range [200, 800], where both left- and right- ends are included.
2. Generate 601 random integers using rand() % 601, the result is a random int in [0, 600].
3. To get a random integer in [200, 800], add 200 to the random integer generated in previous step. Image there is a ball-throwing machine randomly throws a ball ranging from 0 to 600.
   Every time we get a ball, add 200 to its face value. As a result, it is equivalent that the ball-throwing machine generates a random integer from 200 to 800.
4. In general, to generate a random integer in [start, end], where start and end are integers and start <= end, we use the following statement,

```
int num = rand() % (end - start + 1) + start;
```

In the following, we use row and line interchangeably.

Problem 2: Write code, for a given size, which is the maximum number of asterisks in any line and is an odd number, to print a pyramid. Here is a shape of size 3.

```
 *
***
```

Here is a shape of size 5.

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

Answer: please see a complete code. Read comments carefully.

```
//Sample input/output:
//Enter an odd number as size of pyramid: 2
//Not an odd number, re-enter: 4
//Not an odd number, re-enter: 6
//Not an odd number, re-enter: 5
//  *
// ***
//*****

#include <iostream>
using namespace std;

int main()
{
    cout << "Enter an odd number as size of pyramid: ";
    int size;
    cin >> size;

    //ensure the size is an odd number
    while (size % 2 == 0) { // size % 2 == 0 means size is even
        cout << "Not an odd number, re-enter: ";
        cin >> size;
    }

    //The number of asterisks starts from 1,
    //increased by 2 when moving to the next line,
    //stop until we have reached size asterisks.
    //Note that since we start from 1, and increased by 2
    //each time, the number of asterisks in each line
    //(including the last line) must be odd number.
    //That is the reason we need the size to be odd number.

    int numAsts = 1; //number of asterisks in each row
    while (numAsts <= size) {
       //Note that symbols in each row consists of
       //(1) preceding spaces:
       //   spaces before the leftmost asterisk in a line,
       //(2) asterisks,
       //(3) succeeding spaces:
       //   spaces after the rightmost asterisk in a line,
       //Note that the total number of symbols in a row is size,
       //that is, in the shape of pyramid,
       //number of preceding spaces + number of asterisks +
       //number of succeeding spaces = size,
       //due to symmetry,
       //number of preceding spaces = number of succeeding spaces
       //So number of preceding spaces = (size - numAsts) / 2.

       //print spaces for (size - numAsts) / 2 times
       for (int i = 0; i < (size - numAsts)/2; i++)
           cout << " ";

       //print * for numAsts times
       for (int i = 0; i < numAsts; i++)
           cout << "*";
```

```
56
57          //No need to print spaces after the rightmost asterisk,
58          //since it make no difference to our eyesights.
59
60          //Every symbol in the current line is displayed,
61          //except the unnecessary succeeding spaces.
62          //It remains to print a newline, that is,
63          //move to the next line.
64          cout << endl;
65
66          //prepare to move the next row, numAsts is increased by 2
67          numAsts += 2;
68      }
69
70      return 0;
71  }
```

Questions asked

shape of asterisks

1. How come we need to print the `" "` and the endl?

   Answer: (1) When there are preceding spaces before the leftmost asterisk in each line, we need to print them out before displaying asterisks. For example, in the following shapes, preceding spaces cannot be ignored.

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

   or

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

   (2) Once we finish printing all **visible** symbols in a line, we do not need to print out trailing or succeeding spaces after the rightmost asterisk, we need to print out a newline either using `cout << "\n";` or `cout << endl;`, so that we can continue to work on symbols in the next line.

2. How would you determine the number of spaces and how would you implement it? Would it require more than one loop to implement the spaces?

   Answer: Normally there are different types of spaces,

   - preceding spaces: spaces before the leftmost asterisk
   - succeeding spaces: spaces after the rightmost asterisk. We do not need to print these spaces.

- In other patterns, there are spaces between asterisks. It depends the actual patterns and we do not discuss here.

To print spaces in a line, one loop is enough. However, to print spaces in multiple lines, we need an outerloop to calculate the number of spaces in different lines.

We determine the number of spaces by counting in each row. For the example of pyramid of size 5.

␣␣*
␣***
*****

row	number of preceding spaces	number of asterisks
0	2	1
1	1	3
2	0	5

We observe that

- All symbols in a row adds up to size, defined as the maximum of asterisks in any row, that is, 5 in the above example.
  number of preceding spaces + number of asterisks + number of succeeding space = size
- By symmetry, in each row,
  the number of preceding spaces = the number of succeeding spaces
- For each row, we have number of preceding spaces = (size - number of asterisks) / 2.
- Number of asterisks is 1 in the first row, increased by 2 when moving to the next line, until we reach the last line, when the number of asterisks is size.
- The pseudocode of pyramid is as follows.

```
declare and initialize size
declare variable numAsts, representing number of asterisks in each row.
initialize numAsts to be 1, since the number of asterisks in the first row is 1.
while (numAsts is smaller than or equal to size)
begin
   print spaces for (size - numAsts) / 2 times
   print asterisks for numAsts times

   //wrap up current line
   print a newline
   //prepare to move to next line
   increase number of asterisks by 2
end
```

Note that print spaces for x times can be written as

```
for (int i = 0; i < x; i++)
    cout << " ";
```

In more general, pseudocode of doing something for x times can be written as

```
for (int i = 0; i < x; i++)
    doing someting;
```

3. Not sure how to space it in the question above and how to get only 3 rows with size 5.

   Answer: For explanation of calculating spaces in each row, please see the analysis for the above problem.

   As to why there are only 3 rows with size 5, since the number of asterisks is increased by 2 when moving to the next adjacent line. That is, the first row has 1 asterisk, the second row has $1 + 2 = 3$ asterisks, and the third row has $3 + 2 = 5$ asterisks.

   In general, let row number start from 1, and the number of asterisks at the first row is m, then the nth row has $m + (n - 1) * 2$ asterisks. In the example of pyramid of size 5, $m = 1$ and $n = 3$, and the 3rd row has 5 asterisks.

4. Why is `num_asterisks--;` used after `cout << endl;`?

   Answer: You are talking about the following example.

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

   After `cout << endl;`, we finish the current line, and `num_asterisks;` is decreased by 1 when we move from one line to the next, hence `num_asterisks--;`.

   There are two approaches to solve the above problem, one is to trace the change of number of asterisks. The other is find out the relationship between row and column when we place asterisks.

   Approach 1. trace change of number of asterisks. Since there is no space appears before the leftmost asterisk in each row, we do not need to consider preceding spaces in this triangle pattern.

   Assume that we have a triangle pattern of size 5.

row	number of asterisks
1	5
2	4
3	3
4	2
5	1

   Observe that number of asterisks starts from size, decreased by 1 in each row, when number of asterisks is reduced to 0, the program stops. In each row, we print asterisks for number of asterisks times.

   A pseudocode for the above pattern is,

```
Declare and initialize size.
Declare and initialize numAsts to be size.
while (numAsts > 0)
begin
  print asterisks for numAsts times
  print a new line to wrap up current row
  decrease numAsts by 1 to prepare to move to next row
end
```

   An implementation is as follows, see top left triangle pattern.

```
//Draw a top left triangle pattern.
//Sample input/output:
//Enter size for triangle: 5
//*****
//****
//***
//**
//*

#include <iostream>
using namespace std;

int main() {
    cout << "Enter size for triangle: ";
    int size;
    cin >> size;

    int numAsts = size;
    while (numAsts > 0) {
        //print asterisks for numAsts times
        for (int i = 0; i < numAsts; i++)
            cout << "*";

        //wrap up the current row
        cout << endl;
        //prepare numAsts for the next row
        numAsts--;
    }
    return 0;
}
```

We may replace the outer while-statement by for-statement, so both outer and inner loops use for-statement. See code for top left triangle using for-statements.

```
//Draw a top left triangle pattern.
//Sample input/output:
//Enter size for triangle: 5
//*****
//****
//***
//**
//*

#include <iostream>
using namespace std;

int main() {
    cout << "Enter size for triangle: ";
    int size;
    cin >> size;

    for (int numAsts = size; numAsts > 0; numAsts--) {
        //print asterisks for numAsts times
        for (int i = 0; i < numAsts; i++)
            cout << "*";
```

```
22
23          //wrap up the current row
24          cout << endl;
25      }
26      return 0;
27  }
```

Approach 2. Read notes on discussion of nested loops. We can think the top left triangle pattern as row goes from 0 to size -1 and column goes from 0 to size -1, where row and column indices start from 0. Find out the relationship of row and column when asterisks are displayed.

row \column	0	1	2	3	4
0	*	*	*	*	*
1	*	*	*	*	
2	*	*	*		
3	*	*			
4	*				

Denote the asterisks on the forward diagonal of 5 x 5 grid in red color, they have labels (0, 4), (1, 3), (2, 2), (3, 1), and (4, 0). The first coordinate in the parentheses is row index, represented by row, and the second coordinate is column index, represented by col, where $0 <= row < 5$ and $0 <= col < 5$.

Observe row and column indices in the forward diagonal satisfy $row + col == size - 1$. In the above example, size is 5.

All asterisks on the top left triangle satisfy $row+col <= size-1$. Said differently, when $row+col < size$, the cell at (row, col) has an asterisk.

The following code can be found at top right triangle where $row + col < size$.

```
1   //Draw a top left triangle pattern.
2   //Sample input/output:
3   //Enter size for triangle: 5
4   //*****
5   //****
6   //***
7   //**
8   //*
9
10  #include <iostream>
11  using namespace std;
12
13  int main() {
14      cout << "Enter size for triangle: ";
15      int size;
16      cin >> size;
17
18      for (int row = 0; row < size; row++) {
19          for (int col = 0; col < size; col++)
20              //(row + col < size) is the same as
21              //(row + col <= size -1)
22              if (row + col < size)
23                 cout << "*";
24              else cout << " ";
25
26          //after one row, print a newline
27          cout << endl;
28      }
```

```
    return 0;
}
```

A simplified version is as follows. Note that $row + col < size$ can be replaced by $col < size - row$;. The simplified code is as follows, see top left triangle simplified with $col < size - row$.

```
//Draw a top left triangle pattern.
//Sample input/output:
//Enter size for triangle: 5
//*****
//****
//***
//**
//*

#include <iostream>
using namespace std;

int main() {
    cout << "Enter size for triangle: ";
    int size;
    cin >> size;

    for (int row = 0; row < size; row++) {
        for (int col = 0; col < size - row; col++)
            cout << "*";

        cout << endl;
    }

    return 0;
}
```

Here is another example, print triangle shape as follows.

␣␣␣␣*
␣␣␣**
␣␣***
␣****
*****

First, count number of preceding spaces – spaces before the leftmost asterisk – and number of asterisks in each row.

row	num of preceding spaces	num of asterisks	num of preceding spaces + num of asterisks
1	4	1	5
2	3	2	5
3	2	3	5
4	1	4	5
5	0	5	5

Note that num of preceding spaces and num of asterisks are not independent, one can be derived from the other since their sum adds is size. In this example, we can use number of asterisks as loop variable. You can use number of preceding spaces as loop variable as well.

```
declare and initialize size
declare loop variable numAsts, representing number of asterisks
```

```
initialize numAsts to be 1
while (numAsts <= size)
begin
    print spaces for (size - numAsts) times
    print * for numAsts times
    print a newline to wrap up the current row
    increase numAsts by 1, which is the number of asterisks in the next row
end
```

An implementation is shown as follows.

```
1  //Draw bottom right triangle.
2  //sample input/output:
3  //Enter size: 5
4  //    *
5  //   **
6  //  ***
7  // ****
8  //*****
9
10 #include <iostream>
11 using namespace std;
12
13 int main() {
14     cout << "Enter size: ";
15     int size;
16     cin >> size;
17
18     int numAsts = 1;
19     while (numAsts <= size) {
20        //print preceding spaces in the current row
21        for (int i = 0; i < size - numAsts; i++)
22            cout << " ";
23
24        //print asterisks in the current row
25        for (int i = 0; i < numAsts; i++)
26            cout << "*";
27
28        cout << endl; //wrap the current line
29        numAsts++; //prepare for the next row
30     }
31
32     return 0;
33 }
```

Similarly, we can analyze the relationship between row and column for the asterisks in this shape.

row \column	0	1	2	3	4
0					*
1				*	*
2			*	*	*
3		*	*	*	*
4	*	*	*	*	*

Observe the bottom right triangle is just opposite from the top left triangle, except they share the same forward diagonal. Given row as row index and col as column index, we have the following,

- The cells on the forward diagonal satisfy $row + col == size - 1$.
- The cells on the bottom right triangle satisfy $row + col >= size - 1$.

when $row + col >= size - 1$, display asterisks, otherwise, display spaces. Here is an implementation. Code link is display bottom right triangle using row and column.

```
//Draw bottom right triangle.
//sample input/output:
//Enter size: 5
//    *
//   **
//  ***
// ****
//*****
#include <iostream>
using namespace std;

int main() {
    cout << "Enter size: ";
    int size;
    cin >> size;

    for (int row = 0; row < size; row++) {
        for (int col = 0; col < size; col++)
            if (row + col >= size-1)
               cout << "*";
            else cout << " ";

        cout << endl; //print a newline after each row
    }

    return 0;
}
```

The above code can be written as the following,

- When $row + col < size - 1$, that is, when $col < size - 1 - row$, print a space. Note that col starts from zero. Combine the above conditions, when $0 <= col < size - 1 - row$, print a space.
- Otherwise, when $col >= size - 1 - row$, print an asterisk. Note that col has a unreachable upper bound size, so when $size - 1 - row <= col < size$, print an asterisk.

Code link is display bottom right triangle using row and column.

```
//Sample input/output:
//Enter size: 5
//    *
//   **
//  ***
// ****
//*****
#include <iostream>
using namespace std;

int main() {
    cout << "Enter size: ";
    int size;
    cin >> size;
```

```
15
16      for (int row = 0; row < size; row++) {
17          //print spaces when col is in [0, size-1-row).
18          for (int col = 0; col < size -1 - row; col++)
19              cout << " ";
20
21          //print asterisks when col is in [size-1-row, size).
22          for (int col = size -1 - row; col < size; col++)
23              cout << "*";
24
25          cout << endl; //print a newline after each row
26      }
27
28      return 0;
29  }
```

5. I am still confused on the asterisks topic. Like how do I get to be all on 1 side or print in the middle.

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

vs.

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

Code of pyramid is shown in Problem 2.

Code of the second shape is shown in the following. The only difference compared with the code of pyramid shape is the number of preceding spaces before the leftmost asterisk in each line.

```
1   //Sample input/output:
2   //Enter an odd number as size of bottom right triangle, each line has asterisks increased by 2: 2
3   //Not an odd number, re-enter: 4
4   //Not an odd number, re-enter: 6
5   //Not an odd number, re-enter: 5
6   //    *
7   //  ***
8   //*****
9
10  #include <iostream>
11  using namespace std;
12
13  int main()
14  {
15      cout << "Enter an odd number as size of bottom right triangle, each line has asterisks
            increased by 2: ";
16      int size;
17      cin >> size;
18
19      //ensure the size is an odd number
```

```
20      while (size % 2 == 0) { // size % 2 == 0 means size is even
21          cout << "Not an odd number, re-enter: ";
22          cin >> size;
23      }
24
25      //The number of asterisks starts from 1,
26      //increased by 2 when moving to the next line,
27      //stop until we have reached size asterisks.
28      //Note that since we start from 1, and increased by 2
29      //each time, the number of asterisks in each line
30      //(including the last line) must be odd number.
31      //That is the reason we need the size to be odd number.
32
33      int numAsts = 1; //number of asterisks in each row
34      while (numAsts <= size) {
35         //Note that symbols in each row consists of
36         //(1) preceding spaces:
37         //   spaces before the leftmost asterisk in a line,
38         //(2) asterisks,
39         //Note that the total number of symbols in a row is size,
40         //that is, in the shape of pyramid,
41         //number of preceding spaces + number of asterisks = size,
42         //So number of preceding spaces = (size - numAsts).
43
44         //print (size - numAsts) spaces
45         for (int i = 0; i < (size - numAsts); i++)
46             cout << " ";
47
48         //print * for numAsts times.
49         for (int i = 0; i < numAsts; i++)
50             cout << "*";
51
52         //move to the next line.
53         cout << endl;
54
55         //prepare to move the next row, numAsts is increased by 2
56         numAsts += 2;
57      }
58
59      return 0;
60  }
```

6. I need help on how to count by more than one in asterisks and printing random number.

   Answer: for asterisk pattern, please read questions on asterisk pattern. For random number, please read questions on random number.

7. How would I output only odd numbers for any given size?

   Answer: starting from an odd number, say 1, add 2 to each time.

   If you want to ensure the input to be an odd number only, use while loop, read answer to Problem 2.

8. How to know when to use or when writing a condition for a while loop?

   Answer: Inside a loop, you may need to categorize something. For example, read Lab 4

   In Task B, if row index and column index have the same parity, print an asterisk, otherwise, print a space.

In Task C, what is the condition of row and column indices to represent a forward and backward diagonal.

9. What is the significance of the syntax for `static_cast<double>(numHits)`?

   Answer: `static_cast<double>(numHits)` converts `int` varible *numHits* to `double` type.

10. Is there a way to do the triangle asterisks with the while or do-while loop?

    Answer: We can use while or do-while loop to print triangle asterisks. However, we need to use two loops (one nested into the other) because the shape is tabular. Normally for tabular output we need to use nested loops, the outer loop controls row, while the inner loop controls column.

11. I do not really get the range part.

    Answer: For example, if you start numAsts from 1, stop when numAsts reach 5 (included), each time numAsts is increased by 1, use the following for-header.

```
//Print shape as follows.
//*
//**
//***
//****
//*****
for (int numAsts = 1; numAsts <= 5; numAsts++)
    ...
```

    If you start numAsts from 5, stop when numAsts reach 1 (included), each time numAsts is decreased by 1, use the following for-header.

```
//Print shape as follows.
//*****
//****
//***
//**
//*
for (int numAsts = 5; numAsts >= 1; numAsts--)
    ...
```

12. if I have to enter a specific range in a while loop, when should I use (and) in the condition or (or).

    For example, for condition of input must be between 0 and 100. Which one do we use?

    (a) while(num<0 or num>100)

    Or

    (b) while (num<0 and num>100)

    I have been using the or in my condition, but my question would be, when would I use the and operator?

    Answer: Normally we use && for and operator and `||` for or operator. There is no number satisfying $(num < 0 \;\&\&\; num > 100)$.

    We would like to *num* to be in $[0, 100]$, that is, *num* falls into the range of $[0, 100]$. That is, we would like $num >= 0 \&\& num <= 100$.

However, if a user does not enter a number out of the range, that is, *num* is smaller than the left boundary OR *num* is bigger than the right boundary, we would like to re-enter *num*.

An example of using && is, when we read a file with a column called date, when we enter start date and end date, saved in variables start and end, respectively.

```
#include <iostream>
using namespace std;

int main() {
    ifstream fin("fileName.txt");

    if (fin.fail()) {
       cerr << "File cannot be opened." << endl;
       exit(1);
    }

    cout << "Enter start date: ";
    string start;
    cin >> start;

    cout << "Enter end date: ";
    string end;
    cin >> end;

    string date;
    double num;
    while (fin >> date >> num) {
         fin.ignore(MAX_INT, '\n'); //may be omitted if you do not need to omit the rest of
             columns
         if (date >= start && date <= end)
            //TODO: when date is in [start, end],
            //process the data.
            ...
    }

    fin.close();
    return 0;
}
```

13. How do you input multiple lines?

    Answer: work on one line, then move to the next. See how the number of asterisk or spaces change.

14. Is there a way to simplify the for loops for printing shapes? For example, define methods to print characters and spaces?

    Answer: we can define a function to print a given number of characters, which we can give examples in the class of 9/18/23.

15. Is there any method to keep teack of loops, especially when you are making shapes?

    Answer: Yes, you can either do debug through online gdb by posting your code to the server or in Visual Studio Code. Here is debug in online gdb and video for debug in online gdb and visual studio code (first twenty minutes).

    Another way is to use Python Tutor C++. Upload code, then click "Visualize Execution" button in the bottom left.

16. How does the spaces within the for loop work such as `cout << " ";` Is it only used for the for loop or can it work in any function?

    Answer: we use `cout << " ";` inside a for-loop if we want to print several spaces. You can use only one occurrence of `cout << " ";` in the following example (Lab 4, Task B) in Lab 4.

    Task B. Checkerboard
    Write a program checkerboard.cpp that asks the user to input width and height and prints a rectangular checkerboard of the requested size using asterisks and spaces (alternating).

    Example:
    Input width: 11
    Input height: 6

    Shape:

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

    A pseudocode is as follows.

```
for (int row = 0; row < height; row++)
begin
    for (int col = 0; col < width; col++)
        if (row and col have the same parity)
           print "*"
        else print " "

    print a newline
end
```

nested loop

Lab 3

1. For Lab 3 B/C, I enter two dates, how to loop number of day between them? Because the dates is string variable, they could not be subtracted?

2. I have more questions about file reading in C++.

    Answer: string cannot be subtracted, but string can be prepared in dictionary order using <, <=, >, and >=. For example, `"aaa" < "abb"` since `"aaa"` appears before `"abb"` in dictionary.

    Please see the following example in online gdb link to compare "aaa" with "abb".

```
#include <iostream>

using namespace std;

int main() {
   //\" is used inside "..." to avoid the " in \"
   //to match with previous ".
```

```
    //In this way, " is shown.
    //For example, cout << "\"hello\""; print out "hello"
    if ("aaa" < "Aaa")
       cout << "\"aaa\" precedes \"Aaa\" in dictionary" << endl;
    else cout << "\"aaa\" does not precede \"Aaa\" in dictionary" << endl;

    return 0;
}
```

Here is another example to find all names between a given range. online gdb link to find records of students in a given name range.

Note grades.txt are uploaded to the server as well. Its contents are as follows.

Table 1: grades.txt

Name	**Quiz1**	**Quiz2**
Ann	92	75
Bob	86	92
Charles	77	91
Deb	69	76
Ed	91	89
Fred	62	89

```
//Read a tsv (tab separated values) file,
//just read the first two columns,
//print out the data in reversed order,
//ie, the first read, the last print out.

//Sample input/output:
//Enter name of the first student: AAA
//Enter name of the last student: FFF
//Ed 91
//Deb 69
//Charles 77
//Bob 86
//Ann 92

#include <iostream>
#include <fstream> //ifstream
#include <climits> //INT_MAX
using namespace std;

int main()
{
    ifstream fin("grades.tsv");

    if (fin.fail()) {
       cerr << "file cannot be opened for reading." << endl;
       exit(1);
    }

    string junk;
    getline(fin, junk); //skip the first line, column header
```

```
31
32      //Enter names
33      cout << "Enter name of the first student: ";
34      string start;
35      getline(cin, start);
36
37      cout << "Enter name of the last student: ";
38      string end;
39      getline(cin, end);
40
41      string name;
42      int score;
43      const int SIZE = 10;//suppose there are at most 10 students
44          //capacity of a statically declared array is a const
45      string allNames[SIZE];
46      int allScores[SIZE];
47
48      int index = 0; //the first index to write to
49      while (fin >> name >> score && name <= end) {
50          fin.ignore(INT_MAX, '\n'); //ignore the rest of line
51             //we are interested in only the first grade
52          if (name >= start) {
53             allNames[index] = name;
54             allScores[index] = score;
55             index++;
56          }
57      }
58
59      for (int i = index-1; i >= 0; i--)
60          cout << allNames[i] << " " << allScores[i]
61               << endl;
62
63      fin.close(); //close a file when finish reading
64      return 0;
65  }
```

3. More explanation about files. I'm stuck in Lab 4.

   Answer: For explanation of file, please see example to read a tab-separated values format file.

   For hints of Lab 4, please read hints for Lab 4.

array

1. I need to learn more about arrays.
2. Can you explain more about arrays?
3. I would like to go over arrays more.
4. I do not really get array.
5. Can we work on arrays again?
6. Arrays still?

   Answer: we will have Project 2, which is a simplified minesweeper game. We use array and random int there.

random number

1. More practice using rand.
2. More on rand int.

   Answer: we will have Project 2, which is a simplified minesweeper game. We use array and random int there.
3. I still have more questions on random. I do not understand the srand(time(NULL)) in the problem.

   Answer: time(NULL) is current time, more precisely, it returns the number of seconds elapsed since 00:00:00 hours, GMT (Greenwich Mean Time), January 1, 1970. srand(time(NULL)) uses the time when the line of srand(time(NULL)) runs to generate a sequence of random integers.
4. How to use srand?

   Answer: you can use srand(1); or srand(2); where the number inside the parentheses of function srand is the seed to generate a sequence of random number.

   In general, we use srand(time(NULL)); to make sure the sequence of random integers are unique when we run the program. Because time(NULL) represents the time when the statement itself runs, and a program cannot run at exactly the same time in different runnings.

   Without srand(time(NULL)); the sequence of random integers generated is obtained by srand(1).

   Code link is generate three random integers.

```
#include <iostream>
#include <cstdlib> //rand
#include <ctime> //time
using namespace std;

int main() {
    //srand(1);
    srand(time(NULL));
    for (int i = 0; i < 3; i++)
        cout << rand() << endl;
        //rand() returns an int in [0, RAND_MAX]

    return 0;
}
```

5. Why is the random generation so long with the distribution method?

   Answer: a code is shown as follows. I think the name is long, the code itself is not long. You can use answer in Problem 1 if you like.

```
#include <random>
using namespace std;

int main()
{
    std::default_random_engine generator;
    std::uniform_int_distribution<int> distribution(1,6);
    int dice_roll = distribution(generator); // generates number in the range 1..6

    cout << dice_roll << endl;
    return 0;
}
```

Monte Carlo

1. How would you use the above code to create a circle instead?

2. When do we use Monte Carlo?

   Answer: Monte Carlo simulation is a method to estimate π. It does not print a circle inside the square.

   The circle is to illustrate how to estimate π using Monte Carlo simulation. We may think a circle with radius 1 centered at a a square where horizontal coordinate x ranges from -1 to 1 and vertical coordinate y ranges from -1 to 1. The square serves as a dart board.

   Generate random floating point numbers x and y such that $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$.

   (1) Point (x, y) falls into the square all the time.

   (2) However, only when $x^2 + y^2 \leq 1$, point (x, y) falls into the circle.

   (3) The probability that a dart hits the circle is the area of circle divided by the area of square, that is, $\pi/4$.

   (4) The probability that a dart hits the circle can be **estimated** by number of hits on the circle divided by the total number of throws when throwing a dart **sufficient many** times to the dart board.

   (5) Combine (3) and (4), *pi* is estimated as number of hits on the circle divided by the number of total tries, then multiplied by 4.

administration

1. Is recitation only for code review and quizzes? I'm asking this because we barely got help on our labs in the previous recitations.

   Answer: In this semester, we use recitation mainly for lab quiz and code review. If we have more time after finishing lab quiz and code review for every student, we can do tutoring in recitation. Questions given in code review are similar to labs, assignments, and projects that are due before the date when recitation is held. In this way, students can discuss implementation details with lab instructors or TAs. This adds a little human grading in your programs, sometimes you might write codes that pass gradescope, but the implementation can be made better.

2. For tutoring, do we walk in or do we schedule an appointment on navigate?

   Answer: Tutoring is on 11:30 - 5:30 M - F when school is open. Location is North building 1001 B.

   No need to make an appointment, just walk in, it will be in first-come first serve basis.

3. Is there a cheat sheet for the 135 final?

   Answer: Yes, we will provide a cheat sheet as cheat sheet.

4. When is next quiz?

   Answer: around begining of October. I will make an announcement before the quiz.

resources What are good online source other than textbook to learn?
Answer: video: C Programming Tutorial for Beginners from youtube.
Link: C Programming Tutorial for Beginners
book: C Programming: A Modern Approach
book of "C Programming: A Modern Approach".