

# Memory Game

## 1 Introduction

Memory game is a textual implementation of <https://www.helpfulgames.com/subjects/brain-training/memory.html>.

- Randomly place several pairs of integers into slots. To make the game challenging, add empty-value cards.
- In the beginning, all the cards are facedown, not showing their values.
- When flipping a first card, the value of the card is shown.
- When flipping a second card, if its value matches with that of the first card, then the second card is face up, that is, its value is shown; otherwise, both cards are flipped facedown.
- Keep flipping until all the cards with values are uncovered.
- Report the number of rounds needed to reveal all value-bearing cards.

## 2 Problem Description

1. Initialization
  - (a) By default, put 3 pairs of integers, ranging from 0 to 999, together with two empty strings to 8 slots.
  - (b) Use an array of strings to represent the cards since the data includes both strings and integers. We can use `to_string(int num)` to convert integers to strings.
  - (c) We can also define a non-default version, placing *numPairs* pairs of random integers in the range [0, 999] within *numSlots* slots, where *numPairs* and *numSlots* are specified by users.
  - (d) Additionally, we may use words instead of integers as the values of cards.
2. Randomly distribute the strings representing integers and empty strings within the array.
3. Enter an integer representing a valid index of a unflipped card.
4. Display the layout of cards in a tabular format.
5. Play the game.

### 3 Design

Here is header file `MemoryGame.hpp` of `MemoryGame` class, declaring data members and methods of `MemoryGame.cpp`.

`#ifndef ... #define ... #endif` are called include guard. They can make sure a header file will not be included more than once. For example, suppose Hare and Tortoise are inherited from `Animal` class, then `Animal.hpp` is included in both `Hare.hpp` and `Tortoise.hpp`. Then when we have a competition involves Hare and Tortoise, `Animal.hpp` would have been included more than once, as a result, data members and function members in `Animal.hpp` are declared more than once, which causes compilation problem.

```
1 #ifndef _MEMORY_GAME
2 #define _MEMORY_GAME
3 #include <string>
4 using namespace std;
5 class MemoryGame {
6 public: //public method member, any class can use these methods
7     MemoryGame();
8         //default constructor, with 3 pairs of random integers in range [0, 999]
9         //placed in 8 blocks (two blocks are empty).
10    MemoryGame(int numPairs, int numSlots);
11        //Place numPairs pairs of random integers in range [0, 999] in numSlots space,
12        //need numPairs > 0, numSlots > 0, and numSlots >= 2 * numPairs
13    MemoryGame(string *words, int size, int numSlots);
14        //instead of randomly generated integers,
15        //use words as data
16    ~MemoryGame();
17    void play();
18    void display() const;
19        //display array values, if bShown[i] is true,
20        //then values[i] is displayed, where i is the index.
21    void randomize();
22        //randomize the layout of elements in values.
23    int input() const;
24        //input an int that is a valid index and
25        //the corresponding element of values is not shown yet.
26        //That is, the input i is in [0, numSlots) and
27        //bShown[i] is false.
28 private: //private data members, private means that
29         //only methods in this class, not other class,
30         //can access or modify these data members.
31    int numPairs; //numPairs of identical twin items
32    int numSlots;
33        //size of array value, besides identical twins,
34        //may contain empty string to
35        //make the problem more challenging
36    string *values;
37        //a string to represent the layout of data,
38        //mixed with possible empty strings.
39        //Use array to access each element in const time.
40    bool *bShown;
41        //an array of boolean to indicate which element of
```

```

42 //array values is shown or not.
43 //If bShown[i] is true, then values[i] is shown,
44 //otherwise values[i] is not shown,
45 //where 0 <= i and i < numSlots.
46 };
47 #endif

```

## 4 Task A

Implement constructors and destructor.

### 4.1 Define default constructor

1. Set **data member** `numPairs` to be 3.
2. Set **data member** `numSlots` to be 8.
3. Allocate dynamic memory for an array of strings with `numSlots` elements and assign it to **data member** `values`. Generate three random integers in  $[0, 999]$ , place the first integer to the first two slots of `values`, the second integer to the next two slots of `values`, and the third integer to the subsequent two slots of `values`. Set the rest elements of `values` to be empty strings.
4. Allocate dynamic memory for an array of booleans – type `bool` in C++ – with `numSlots` elements and assign it to **data member** `bShown`. Set each element of `bShown` to be false.

### 4.2 Non-default constructor `MemoryGame(int numPairs, int numSlots)`

1. If at least one of formal parameters `numPairs` and `numSlots` is non-positive or `numSlots` is smaller than twice of `numPairs`, then set **data members** `numPairs` to be 3 and **data members** `numSlots` to be 8.  
Otherwise, set **data member** `numPairs` by formal parameter `numPairs`, and **data member** `numSlots` by formal parameter `numSlots`.
2. Allocate dynamic memory for an array of strings with `numSlots` elements and assign it to **data member** `values`. Generate `numPairs` random integers in  $[0, 999]$ , place the first integer to the first two slots of `values`, the second integer to the next two slots of `values`,  $\dots$ , until all integers are placed in pairs in `values`. Set the rest elements of `values` to be empty strings.
3. Allocate dynamic memory for an array of booleans – type `bool` in C++ – with `numSlots` elements and assign it to **data member** `bShown`. Set each element of `bShown` to be false.

### 4.3 Non-default constructor `MemoryGame(string* words, int size, int numSlots)`

- Assume the number of elements in formal parameter `words` is `size`. If formal parameters `size` or `numSlots` is non-positive or `numSlots` is smaller than twice of `size`, set `numSlots` to be twice of `size`.
- After the above possible adjustment, set **data member** `numPairs` to be formal parameter `size`, and set **data member** `numSlots` to be formal parameter `numSlots`.
- Allocate dynamic memory for an array of strings with `numSlots` elements and assign it to **data member** `values`. Place the first word to the first two slots of `values`, the second word to the next two slots of `values`,  $\dots$ , until all words are placed in pairs in `values`. Set the rest elements of `values` to be empty strings.

- Allocate dynamic memory for an array of booleans – type `bool` in C++ – with `numSlots` elements and assign it to **data member** `bShown`. Set each element of `bShown` to be false.

## 4.4 Define destructor

Release dynamically allocated memory allocated to data member `values`. Set `values` to be `nullptr` to avoid dangling pointer problem. Do similar things to data member `bShown`.

## 4.5 What you need to do

Define `MemoryGame.cpp`. Note that there are some differences in defining methods, aka member functions, in a class and a function that is not associated with a class.

- If the header file and source codes are separated as in our case, need to add `className` followed by `::` before a method of class. For example, define default constructor `MemoryGame` in `MemoryGame.cpp`, we use the following code snippet.

```
1 MemoryGame::MemoryGame() {
2     //TODO: Your code goes here.
3 }
```

- A constructor of a class is the creator of an object in that class. It must have exactly the same name as a class, case to case, letter to letter. However, a constructor has no return type, not even `void`.

We can have multiple versions of constructors, as long as their signatures, i.e., parameter lists, are different. These distinctions can be in terms of the number of parameters, the types of parameters, or the order of parameters.

- We can imagine a class as a branch concentrating on manufacturing a product, with data members akin to items placed within the lounge of that branch. Constructors and methods within a class can directly access and modify these data members. However, functions outside the class can only access or modify the data members through the public methods provided by that class.

As a result, no need to pass data members as parameters or return their values in constructors or methods of the same class.

```
1 #include "MemoryGame.hpp"
2
3 MemoryGame::MemoryGame() {
4     //TODO: place your code here
5 }
6
7 MemoryGame::MemoryGame(int numPairs, int numSlots) {
8     //TODO: place your code here
9 }
10
11 MemoryGame::MemoryGame(string* words, int size, int numSlots) {
12     //TODO: place your code here
13 }
14
15 MemoryGame::~MemoryGame() {
```

```

16 //TODO: place your code here
17 }
18
19 void MemoryGame::randomize() {
20     //TODO: placeholder, do not need to implement in Task A
21 }
22
23 void MemoryGame::display() const {
24     //TODO: placeholder, do not need to implement in Task A
25 }

```

**Warning:** remove all instances of `srand(time(NULL));` statement before submitting to gradescope.

## 4.6 Reduce code redundancy

You may find out there are a lot of redundant codes in these constructors. One way is to define `MemoryGame(int numPairs, int numSlots)`, then use constructor delegator to simplify the definition of other two constructors. For example, default constructor can be defined as follows with the help of non-default constructor `MemoryGame(int, int)`.

```

1 MemoryGame::MemoryGame() : MemoryGame(3, 8) {
2 }

```

Warning: to use constructor delegate, we need C++11. So when running `g++` command, add `-std=c++11` option.

## 4.7 Test your code locally

Download `MemoryGame.hpp` from blackboard. Define `MemoryGame.cpp` and Define `MemoryGameClient.cpp` with the following contents.

```

1 #include <iostream>
2 #include <string>
3 #include "MemoryGame.hpp"
4 using namespace std;
5
6 int main() {
7     MemoryGame game;
8     game.~MemoryGame();
9
10    MemoryGame game2(3, 9);
11    game2.~MemoryGame();
12
13    string words[] = {"Hello", "Hi", "Hey"};
14    int size = sizeof(words) / sizeof(words[0]);
15    MemoryGame game3(words, 3, 9);
16    game3.~MemoryGame();
17
18    return 0;
19 }

```

Here are the commands to compile the codes. Note that we have two C++ source codes: MemoryGame.cpp and MemoryGameClient.cpp.

```
g++ -std=c++11 -o run MemoryGameClient.cpp MemoryGame.cpp
./run
```

Explanation: `-std=c++11` means to use C++11. `-o run` means to output a runnable file called run. If we use `-o memory`, then the runnable file is called memory.

Your code should run without errors. At this step, there is no output yet.

## 4.8 Use makefile

A better way to run a project with multiple source codes is to use `makefile`, created for the project.

In this case, if a file is modified, `make` command only compiles the modified code and relink the object file created by this file. No need to recompile and relink unchanged files.

```
1 # This is an example Makefile for Memory Game project. This
2 # Typing 'make' or 'make run' will create the executable file.
3
4 # define some Makefile variables for the compiler and compiler flags
5 # to use Makefile variables later in the Makefile: $()
6 #
7 # -g adds debugging information to the executable file
8 # -Wall turns on most, but not all, compiler warnings
9 #
10 # for C++ define CC = g++
11 CC = g++ -std=c++11
12
13 #CFLAGS = -g -Wall
14
15 # typing 'make' will invoke the first target entry in the file
16 # (in this case the default target entry)
17 # you can name this target entry anything, but "default" or "all"
18 # are the most commonly used names by convention
19 #
20 all: run
21
22 # To create the executable file run we need the object files
23 run: MemoryGameClient.o MemoryGame.o
24     $(CC) -o run MemoryGameClient.o MemoryGame.o
25
26 # To create the object file MemoryGameClient.o,
27 # we need source MemoryGameClient.cpp
28 MemoryGameClient.o: MemoryGameClient.cpp
29     $(CC) -c MemoryGameClient.cpp
30
31 # To create the object file MemoryGame.o, need source file MemoryGame.cpp
32 MemoryGame.o: MemoryGame.cpp
33     $(CC) -c MemoryGame.cpp
34
```

```

35 clean:
36 $(RM) run *.o *~

```

Then run the following commands.

```

make
./run

```

## 5 Task B: complete method randomize

In this task, we will randomize the elements in data member array **values**, whether the elements are numbers or strings.

Based on the code of Task A, define **randomize** function. The idea is similar to the randomization process of Project 1 Task D.

1. Initialize **size** to be the number of elements of array **values**, which is saved in data member ... (you find this out).
2. Generate a random index in  $[0, \text{size})$ . Swap the element at chosen index with the last element in **values**.
3. Then reduce **size** by 1 so that we do not consider the already-chosen element, which resides in the last element of **values**.
4. Continue Step 2-3 until **size** is 1 (why not 0? Hint: do we need to randomize if there is only one element left in the array to be randomized?) Said differently, as long as **size** is larger than 1, run codes in Steps 2-3.

## 6 Task C: implement display method

You may want to use the following code to draw seperated lines.

```

1 void printSeparatedLine(int size) {
2     cout << "+"; //the first +
3     //draw -----+ for (size) many times
4     for (int i = 0; i < size; i++)
5         cout << "-----+";
6
7     cout << endl;
8 }

```

The following is a piece of code to print out the label.

```

1 //print labels
2 cout << " ";
3 for (int i = 0; i < numSlots; i++)
4     cout << setw(3) << i << setw(3) << " ";
5     //setw(3) before i means i occupies 3-character,
6     //For example, if i has only two digits,
7     //then pad a space to its left.
8
9 cout << endl;

```

A sample output is shown as follows. Note that each number is right aligned in a 5-character column. You can use `setw(5)` from `iomanip` library to specify that 5 characters are reserved to display a number. There are also 3 spaces before and after each line.

```

      0      1      2      3      4      5      6      7
+-----+-----+-----+-----+-----+-----+-----+
| 886 | 777 | 383 | 777 | 383 | 886 |
+-----+-----+-----+-----+-----+-----+

```

You do not need to call `randomize` method in constructors. So, at this point, the elements in data member `values` are not randomized yet.

We will call `randomize` method in `play` method, which is required in Task D.

## 7 Task D: implement input and play methods

It remains to implement input method and play method.

### 7.1 implement input method

Ensure users to enter an integer represented a valid index of a unflipped card.

1. Start to print the following prompt. Warning: there is a space following :  
Enter a unflipped card in `[0, last_index_of_data_member_values]`:
2. If users enter an integer that is not a valid index, that is, either is negative or larger than the last index of data member `values`, print the following prompt. Warning: there is a space following :  
input is not in `[0, last_index_of_data_member_values]`. Re-enter: (followed by a space)
3. If users enter the index of a card is flipped already, print out prompt “The card is flipped already. Re-enter: ” without quotes.
4. Let user re-enter if the previous input is not valid.

### 7.2 implement play method

## 8 A sample run of the code

Caution: before running the code, remove all occurrences of `srand(time(NULL));` statements in `MemoryGame.cpp`.

The following output runs in [https://www.onlinegdb.com/online\\_c++\\_compiler](https://www.onlinegdb.com/online_c++_compiler).

Please note that the random sequence generated on a Mac may differ from that on a Linux server. Gradescope and onlinegdb operate on a Linux server.

### 8.1 Test play method of an object created by default constructor

The content of main function is as follows.

```

1  MemoryGame game;
2  game.play();

```

Here is a sample run.



0	1	2	3	4	5	6	7

Round 1:

Enter a unflipped card in [0, 7]: -1  
input is not in [0, 7]. Re-enter: 10  
input is not in [0, 7]. Re-enter: 0

0	1	2	3	4	5	6	7
886							

Round 2:

Enter a unflipped card in [0, 7]: -3  
input is not in [0, 7]. Re-enter: 0  
The card is flipped already. Re-enter: 1

0	1	2	3	4	5	6	7

Round 3:

Enter a unflipped card in [0, 7]: 1

0	1	2	3	4	5	6	7
	777						

Round 4:

Enter a unflipped card in [0, 7]: 2

0	1	2	3	4	5	6	7

Round 5:

Enter a unflipped card in [0, 7]: 2

0	1	2	3	4	5	6	7

Round 6:

Enter a unflipped card in [0, 7]: 3

0	1	2	3	4	5	6	7

Round 7:

Enter a unflipped card in [0, 7]: 3

0	1	2	3	4	5	6	7
			383				

```

+-----+-----+-----+-----+-----+-----+-----+
Round 8:
Enter a unflipped card in [0, 7]: 4
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+
|       |       |       |       |       |       |       |
+-----+-----+-----+-----+-----+-----+-----+
Round 9:
Enter a unflipped card in [0, 7]: 4
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+
|       |       |       |       | 777|       |       |
+-----+-----+-----+-----+-----+-----+-----+
Round 10:
Enter a unflipped card in [0, 7]: 1
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+
|       | 777|       |       | 777|       |       |
+-----+-----+-----+-----+-----+-----+-----+
Round 11:
Enter a unflipped card in [0, 7]: 5
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+
|       | 777|       |       | 777|       |       |
+-----+-----+-----+-----+-----+-----+-----+
Round 12:
Enter a unflipped card in [0, 7]: 6
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+
|       | 777|       |       | 777|       |       |
+-----+-----+-----+-----+-----+-----+-----+
Round 13:
Enter a unflipped card in [0, 7]: 6
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+
|       | 777|       |       | 777|       | 383|       |
+-----+-----+-----+-----+-----+-----+-----+
Round 14:
Enter a unflipped card in [0, 7]: 3
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+
|       | 777|       | 383| 777|       | 383|       |
+-----+-----+-----+-----+-----+-----+-----+
Round 15:
Enter a unflipped card in [0, 7]: 0
  0     1     2     3     4     5     6     7
+-----+-----+-----+-----+-----+-----+-----+
| 886| 777|       | 383| 777|       | 383|       |
+-----+-----+-----+-----+-----+-----+-----+

```

Round 16:

Enter a unflipped card in [0, 7]: 7

```

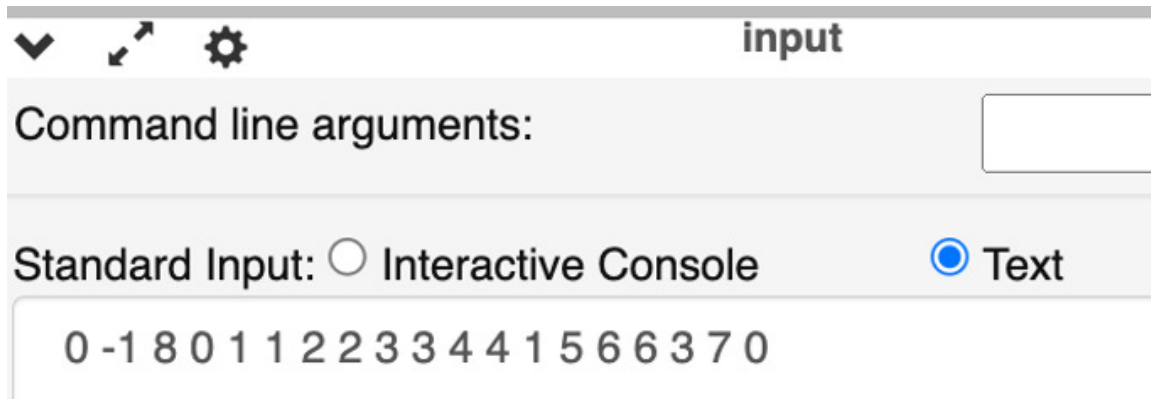
  0    1    2    3    4    5    6    7
+-----+-----+-----+-----+-----+-----+-----+-----+
| 886| 777|     | 383| 777|     | 383| 886|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Congratulations! Found out all pairs in 16 rounds

You can also use text input instead of interactive input in onlinegdb. In **input** pane, under **Standard input:**, click **Text** radio box. Then enter the following inputs in text box. Click Run button.

0 -1 8 0 1 1 2 2 3 3 4 4 1 5 6 6 3 7 0



If your code works fine, your would see the following output.

```

  0    1    2    3    4    5    6    7
+-----+-----+-----+-----+-----+-----+-----+-----+
|      |      |      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Round 1:

Enter a unflipped card in [0, 7]: 0 1 2 3 4 5 6 7

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| 886|      |      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Round 2:

Enter a unflipped card in [0, 7]: input is not in [0, 7]. Re-enter: input is not in [0, 7]. Re-enter:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|      |      |      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Round 3:

Enter a unflipped card in [0, 7]: 0 1 2 3 4 5 6 7

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|      | 777|      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

...

Round 15:

Enter a unflipped card in [0, 7]: 0 1 2 3 4 5 6 7

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|      | 777|      | 383| 777|      | 383| 886|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

```

+-----+-----+-----+-----+-----+-----+-----+
Round 16:
Enter a unflipped card in [0, 7]:    0      1      2      3      4      5      6      7
+-----+-----+-----+-----+-----+-----+-----+
| 886| 777|      | 383| 777|      | 383| 886|
+-----+-----+-----+-----+-----+-----+
Congratulations! Found out all pairs in 16 rounds

```

## 8.2 Test code with non-default constructor with words

A sample run of the code with words “Hello” and “Hi” placed in 7 slots. The code in main method is as follows.

```

1  string words[] = {"Hello", "Hi"};
2  int size = sizeof(words) / sizeof(words[0]);
3  MemoryGame game(words, size, 7);
4  game.play();

```

```

      0      1      2      3      4      5      6
+-----+-----+-----+-----+-----+-----+-----+
|      |      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
Round 1:
Enter a unflipped card in [0, 6]: 0
      0      1      2      3      4      5      6
+-----+-----+-----+-----+-----+-----+-----+
|      |      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
Round 2:
Enter a unflipped card in [0, 6]: 1
      0      1      2      3      4      5      6
+-----+-----+-----+-----+-----+-----+-----+
|      |      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
Round 3:
Enter a unflipped card in [0, 6]: 1
      0      1      2      3      4      5      6
+-----+-----+-----+-----+-----+-----+-----+
|      |Hello|      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
Round 4:
Enter a unflipped card in [0, 6]: 2
      0      1      2      3      4      5      6
+-----+-----+-----+-----+-----+-----+-----+
|      |      |      |      |      |      |      |
+-----+-----+-----+-----+-----+-----+-----+
Round 5:
Enter a unflipped card in [0, 6]: 2
      0      1      2      3      4      5      6
+-----+-----+-----+-----+-----+-----+-----+

```

```

|   |   |   |   |   |   |
+---+---+---+---+---+---+
Round 6:
Enter a unflipped card in [0, 6]: 3
  0   1   2   3   4   5   6
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
Round 7:
Enter a unflipped card in [0, 6]: 3
  0   1   2   3   4   5   6
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
Round 8:
Enter a unflipped card in [0, 6]: 4
  0   1   2   3   4   5   6
+---+---+---+---+---+---+
|   |   |   |   |   |   |
+---+---+---+---+---+---+
Round 9:
Enter a unflipped card in [0, 6]: 4
  0   1   2   3   4   5   6
+---+---+---+---+---+---+
|   |   |   |   | Hi|   |
+---+---+---+---+---+---+
Round 10:
Enter a unflipped card in [0, 6]: 6
  0   1   2   3   4   5   6
+---+---+---+---+---+---+
|   |   |   |   | Hi|   | Hi|
+---+---+---+---+---+---+
Round 11:
Enter a unflipped card in [0, 6]: 1
  0   1   2   3   4   5   6
+---+---+---+---+---+---+
|   |Hello|   |   | Hi|   | Hi|
+---+---+---+---+---+---+
Round 12:
Enter a unflipped card in [0, 6]: 5
  0   1   2   3   4   5   6
+---+---+---+---+---+---+
|   |Hello|   |   | Hi|Hello| Hi|
+---+---+---+---+---+---+
Congratulations! Found out all pairs in 12 rounds

```

You can also use text input instead of interactive input. In **input** pane, under **Standard input:**, click **Text** radio box. Then enter the following inputs in text box. Click Run button.

```
0 1 1 2 2 3 3 4 4 6 1 5
```