COMP 2011 Midterm - Spring 2017 – HKUST

- Date: March 21, 2017 (Tuesday)
- Time Allowed: 2 hours, 7:00 pm - 9:00 pm
- Instructions:
    1. This is a closed-book examination.
    2. There are 9 questions on 16 pages (including the cover page).
    3. Write your answers in the space provided in black/blue ink. NO pencil please.
    4. All programming codes in your answers must be written in ANSI C++.
    5. You may use only the C++ language features and constructs learned in lectures so far. For example, no pointers, C++ classes, string class, etc.
    6. For programming questions, you are NOT allowed to define additional helper functions or structures, nor global variables unless otherwise stated. You also cannot use any library functions not mentioned in the questions.

Student Name	**Mr. Model Answer**
Student ID	
ITSC email	
Lecture and Lab Section	

I have not violated the Academic Honor Code in this examination (signature):_____

Problem	**Score / Max. score**
1. Basics	/10
2. Control Flow	/6
3. Scope	/4
4. Function Overloading	/8
5. Recursion	/12
6. C String & Recursion	/10
7. Multi-dimensional Array	/13
8. Array	/15
9. Programming	/22
Total	**/100**

Problem 1 C++ Basics *[10 marks]*

```
int a, b, c = 2, d, e, p = 5, q = 1;
a = p / 2 + 1;
b = (++a);
d = c += (b++);
e = p + (p == q);
```

Evaluate the values of the following variables after the execution of the above code segment:

Variable Name	p	q	a	b	c	d	e
Value	5	1	4	5	6	6	5

Grading scheme: 2 marks each value

Problem 2 Control Flow *[6 marks]*

(a) [2 marks] What is the output of the following program?

```
int j = 0;
for(; j<3; ++j);
cout << j;
cout << endl;
```

Answer: ___3 _____

(b) [2 marks] What is the output of the following program?

```
int k = 0;
do
{
   ++k;
   if (k != 0) continue;
} while(k < 3);
cout << k << endl;
```

Answer: ___3 _____

(c) [2 marks] What is the output of the following program?

```
int p = 0;
while(p <= 3)
{
   p++;
   if (p != 0) break;
}
cout << p << endl;
```

Answer: ___1_____

Problem 3 Scope *[4 marks]*

What is the output of the following program?

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

int x = 10;

void mystery(int x, int& y)
{
   for (; y < 10; y++)
   {
      int y = -1;
      y++;
   }
   x++;
}

int main()
{
   int y = 3;
   x++;
   mystery(x, y);
   cout << x << endl;
   cout << y << endl;
}
```

11

Answer: _____10_____

Grading scheme: 2 marks each value, total 4 marks (if not on separate lines, -0.5 mark)

Problem 4 Function Overloading *[8 marks]*

Given the following function definitions:

```
void mystery() { cout << 0 << endl; }

void mystery(int& x) { cout << x << endl; }

void mystery(int x, double& y, double z = 6.5)
{
   cout << (100 + x + y + z) << endl;
}

void mystery(double x, int y, double z = 10)
{
   cout << (x + y + z) << endl;
}
```

And the following variable definitions and initializations:

```
int a = 4;
double b = 2.2;
double& c = b;
```

Determine whether the following function call statements are valid or not. A statement is valid if it is syntactically correct and can be compiled and executed without errors.

Statement	Is it valid?	If yes, what is the output?
`mystery();`	Yes	0
`mystery(10);`	No	
`mystery(c, a);`	Yes	16.2
`mystery(b, c);`	No	
`mystery(c, a, b);`	Yes	8.4

Grading scheme: 2 marks each row:

- Both "Yes" and output value correct – 2 marks, otherwise, "Yes" correct but output incorrect – 1 mark
- Wrongly answered "Yes" as "No" – 0 marks

Problem 5 Recursion *[12 marks]*

Consider the following functions

```
int a(int n)
{
   if (n < 0)
      return -n;
   if (n == 0)
      return 100;
   else
      return ( a(n-1) + a(n-2) );
}

int b(int n)
{
   if (n < 0)
      return 1;
   else
      return ( b(n-1) + c(n-1) );
}

int c(int n)
{
   if (n <= 1)
      return 2;
   else
      return ( b(n/2) + c(n/2) );
}
```

You can assume that all the above functions are declared before being called.

Fill the table below, which shows the value of **a(n)**, **b(n)**, and **c(n)** when **n** is 1, 2, 3, or 4.

	When **n** is **1**	When **n** is **2**	When **n** is **3**	When **n** is **4**
Value of **a(n)**	101	201	302	503
Value of **b(n)**	5	7	14	21
Value of **c(n)**	2	7	7	14

Grading scheme: 1 mark each value

Problem 6 C String and Recursion *[10 marks]*

(a) [5 marks] Implement a recursive function, **`cstring_length`**, which returns the length of a C string, **`str`**, starting at the index, **pos**. You can assume that **pos** is greater or equal to 0 and is always smaller than the length. For example, the function call `cstring_length("c++", 0)` returns 3. The function call `cstring_length("c++", 1)` returns 2. The function call `cstring_length("c++ is a piece of cake", 0)` returns 22. The function call `cstring_length("", 0)` returns 0.

```
int cstring_length(char str[], int pos)
{
   if (str[pos] == '\0')    // base case correct: 2 marks or nothing
      return 0;
   else                     // recursive case correct: 3 marks or nothing
      return 1 + cstring_length(str, pos+1);
}
```

(b) [5 marks] Similarly, implement a recursive function, **`cstring_copy`**, which copies the C string in the **source** array to the **target** array starting at the index, **pos** for both arrays. You can assume that **pos** is greater or equal to 0 and is always smaller than the length of the C string in the **source** array. You can also assume that the size of the **target** array is equal or larger than the **source** array. For example:

```
char string1[100] = "c++ is a piece of cake", string2[100] = "funny";
cstring_copy(string2, string1, 0);
cout << "The output is:" << string1 << "." << endl;
```

the above segment of code will output:

```
The output is:funny.
```

```
void cstring_copy(const char source[], char target[], int pos)
{
   target[pos] = source[pos];   // correctly assigning a null character
                                // to target: 1 mark
   if (source[pos] == '\0')     // base case: 2 marks or nothing
      return;
   else                         // recursive case: 2 marks or nothing
      cstring_copy(source, target, pos+1);
   /* Or:
   if (source[pos] == '\0')
      target[pos] = '\0';
   else {
      target[pos] = source[pos];
```

```
        cstring_copy(source, target, pos+1);
    }
    */
}
```

Problem 7 Multi-dimensional Array *[13 marks]*

(a) [2 marks] What is the output of the following code segment?

```
int a[3][4] = { {1, 2, 3, 4}, {5, 6, 7}};
cout << a[1][1] << a[2][3];
```

Answer: ____60____ // 1 mark each correct number

(b) [3 marks] What is the output of the following code segment?

```
int b[3][4][2] = { {1, 2, 3, 4}, {5, 6, 7}};
cout << b[0][1][1] << b[1][1][0] << b[1][1][1];
```

Answer: ____470____ // 1 mark each correct number

(c) [8 marks] Fill in the blanks below to complete the code that prints all the characters in the **array**. The final output should be exactly "**abcdefgh**" (without the double quotes). There are no newlines or spaces in the output. You need to follow the comments and you cannot change the code or add code outside of the 4 blanks.

```
char array[][2][2] = { { {'a', 'b'} }, { {'c', 'd'} },
                       { {'e', 'f'} }, { {'g', 'h'} } };
int i=1;
do
{
   int j = 0;
   while ( _________!j_________ )  //OR   j-1
      // you must not use "==", "!=", "<=", ">=", "<", or ">"
      // in this condition
   {
      for ( _________int k=0; k<2; k++ _________ )
          // the control variable for this loop must be named k
      {
         cout << array[ _________ i - 1 _________ ][j][k];
      }
      j++;
   }
   i++;

} while ( _________i <= 4_________ );
    //you must use the variable i in this condition
```

Grading scheme: 2 marks or nothing for each blank

Problem 8 Array *[15 marks]*

Suppose you are Prof. Horner. All the answer sheets of COMP2011 midterm have been graded and students have been enquiring about the **Mean** and **Standard Deviation** of the midterm. Normally, you would ask the TAs to do the calculations. This year, however, the TAs are passionate Democrats and they are on strike to protest against the Donald Trump-domination of the course. Therefore, you are on your own. Before the TAs go on strike, they responsibly left you a note saying,

> **"Midterm absentees have been given a score of -1. Don't include them when calculating course statistics".**

Implement a function

```
void calculateStatistics(const int scores[], int num,
        double& mean, double& standardDev);
```

to calculate the mean and standard deviation, and return them through the reference variables, **`mean`** and **`standardDev`**. The array, **`scores`**, contains the midterm examination scores of all the students and the variable, **`num`**, is the size of the **`scores`** array.
You should **NOT** include the scores of absent students in the calculation of the mean and the standard deviation. You can use the following library function

```
double sqrt(double x);
```

which returns the square root of **`x`**.

The formula for Standard Deviation is:

$$\sigma = \sqrt{\frac{Variance}{N}}$$

where

$$Variance = \sum_{i=1}^{N} (x_i - \mu)^2$$

where μ is the mean, σ is the standard deviation, N is the total number of students **attended** the midterm examination; x_i is the score of the i^{th} attended student.

```
void calculateStatistics(const int scores[], int num,
                         double& mean, double& standardDev)
{
   // Add your code here for calculating mean

   double sum = 0;
   int validScores = 0;
   for (int i=0; i < num; i++)
   {
      if (scores[i] != -1)
      {
         validScores++;
         sum = sum + scores[i];
      }
   }
   mean = sum/validScores;

   /* grading scheme: correct mean calculation - 4 marks or nothing
                      excluding absentees - 2 marks or nothing */


   // Add your code here for calculating standard deviation

   double variance = 0;
   for (int i = 0; i < num; i++)
   {
      if (scores[i] != -1)
         variance = variance +
                    ((scores[i] - mean) * (scores[i] - mean));
   }
   standardDev = sqrt(variance/validScores);

   /* grading scheme: correct variance calculation - 4 marks or nothing
            correct standard deviation calculation - 3 marks or nothing
                                 excluding absentees - 2 marks or nothing */

}
```

Problem 9 Programming *[22 marks]*

The Hong Kong Chief Executive election 2017 is upcoming. You are asked to complete the functions of an incomplete C++ program to simulate the election.
You are given the following code and you need to define two functions.

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

const int MAX_SIZE = 20;

// Given
void sortArrayOfInt(int arr[], int n);

// Part a
void randomizeArrayOfSum(int arr[], int n, int s);

// Part b
int printResult( /* To be completed */ );

int main()
{
    const int NUM_OF_CANDIDATES = 3, NUM_OF_VOTES = 1194;
    const char BAR_CHAR = '*';
    const char CANDIDATE_NAMES[NUM_OF_CANDIDATES][8] = {
    "John", "Carrie", "Woo"};

    srand(time(NULL));

    int vote_array[NUM_OF_CANDIDATES];
    cout << "Hong Kong Chief Executive Election 2017 simulator" << endl;

    randomizeArrayOfSum(vote_array, NUM_OF_CANDIDATES, NUM_OF_VOTES);

    int electedIndex = printResult(CANDIDATE_NAMES, vote_array,
                NUM_OF_CANDIDATES, BAR_CHAR);

   cout << CANDIDATE_NAMES[electedIndex] << " has won the election."
          << endl;

   return 0;
}
```

(a) [10 marks] To simulate the election, your program will first ***randomly*** assign the votes to the all candidates. For example, if the total number of votes is 1194, it is equivalent to making an array of 3 random integer elements that sums up to 1194, for example, {102, 555, 537}. You are required to write a general function with the prototype:

```
void randomizeArrayOfSum(int arr[], int n, int s);
```

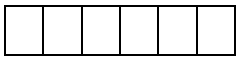
where **n** is the size of the array, **arr**, and **s** is the required sum. You may assume **n** is always smaller than the global integer constant **MAX_SIZE**.

The function will ***assign all the*** **n** ***elements of the array,*** **arr**, ***so that their sum is equal to*** **s** by the following steps. Consider the function call:

```
randomizeArrayOfSum(x, 5, 20);
```

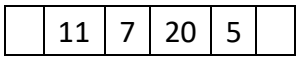
where **x** is an array of 5 integers.

Step 1: A temporary array of size **MAX_SIZE** is created. In the subsequent steps, only the first **(n+1)** elements of the temporary array will be used.

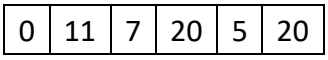


Step 2: For the temporary array, assign random integers of range from 0 to **s** **inclusively** (i.e., [0, **s**]) from the 2nd element to the n-th element. You can use the library function:

```
int rand();
```



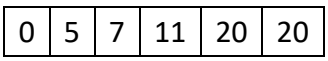
Step 3: Assign the 1st element to be 0 and the (n+1)-th element to be **s**



Step 4: Sort the array in ascending order. Note: You DO NOT need to implement the sorting part. You are given a helper function:

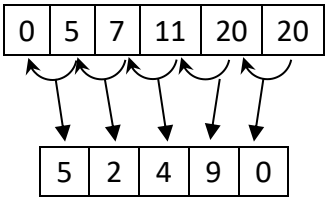
```
void sortArrayOfInt(int arr[], int n);
```

which sorts the element of an **n**-size array in ascending order:



Step 5: Iterate through all the elements of the resultant array.

In each iteration, subtract the i-th element in the temporary array from the (i + 1)-th element in the temporary array and assign the result to the i-th element of the resultant array.



As a result, the array, **x**, will become {5, 2, 4, 9, 0}, that all the elements sum up to exactly 20.

Complete the function **randomizeArrayOfSum** in the box below.

```
void randomizeArrayOfSum(int arr[], int n, int s)
{
   // Step 1: Create an array of size MAX_SIZE
   int temp[MAX_SIZE];    // 1 mark or nothing

   // Step 2:  Assign n-1 random integers ([0, s]) to the array
   // starting at the second element
   for (int i = 1; i < n; ++i) {    // 3 marks or nothing
      temp[i] = rand() % (s + 1);
   }



   // Step 3: assign 0 to the first element and s to the (n+1)-th element

   temp[0] = 0;    // 1 mark or nothing
   temp[n] = s;    // 1 mark or nothing


   // Step 4: sort the array by calling sortArrayOfInt()
   sortArrayOfInt(temp, n + 1);    // 2 marks or nothing


   // Step 5: assign values to each element of arr
   for (int i = 0; i < n; ++i) {    // 2 marks or nothing
      arr[i] = temp[i+1] - temp[i];
   }







}
```

(b) [12 marks] The program will then output election quantitative result by calling the function, **printResult**. You are given the following incomplete function header:

```
int printResult(const char names/* to be completed */,
                const int arr[], int n, char symbol)
```

where the function has only four parameters. The first parameter is a 2-dimensional character array, **names**, containing the names of the candidates; the constant integer array, **arr**, consists of the votes for each candidates; **n** is the number of candidates and **symbol** is the character for printing the bar chart.

The function will illustrate the numbers of votes in the array, **arr**, by a bar chart. The bar chart will have **n** bars. Each bar of votes is represented ***by repeatedly printing the character in the variable,*** **symbol**. Each character represents **20** votes. The character will not be printed if there are less than 20 votes, for example, 8 such characters will be printed for 176 votes. The bar is followed by the **candidate name** and the **number of votes** in "name:#" format. For example,

```
**********Andrew:201
```

The function will also find the candidate who has won the election with the maximum number of votes, and return its corresponding index, e.g. return 1 if the 2nd element in **arr** is the maximum.

The sample outputs of THREE separate executions of the **whole program** are shown below.

```
Hong Kong Chief Executive Election 2017 Simulator
*****John:102
**************************Carrie:555
*************************Woo:537
Carrie has won the election.
```

```
Hong Kong Chief Executive Election 2017 Simulator
************************************John:753
************Carrie:259
*********Woo:182
John has won the election.
```

```
Hong Kong Chief Executive Election 2017 Simulator
***John:73
**************Carrie:285
****************************************Woo:836
Woo has won the election.
```

Complete the function header and the implementation of the **printResult** function in the box below. You are suggested to use a nested loop.

```
// Fill in the blank
int printResult(const char names_______[][8]_______,
                const int arr[], int n, char symbol)
{
   int maxIndex = 0;

   /* grading scheme:
      correct 1st parameter - 2 marks or nothing
      correct output of symbols: 4 marks or nothing
      correct name:#votes - 2 marks or nothing
      correct winner – 3 marks or nothing
      return winner – 1 mark or nothing
   */

   // Add your code here
   for (int i = 0; i < n; ++i)
   {
      for (int j = 0; j < arr[i] / 20; ++j)
      {
         cout << symbol;
      }
      cout << names[i] << ":" << arr[i] << endl;
      if (arr[i] > arr[maxIndex])
        maxIndex = i;
   }
   return maxIndex;

}
```

/* Rough work — You may detach this page */

/* Rough work — You may detach this page */