COMP 2011 Midterm - Fall 2013 - HKUST

Date: Tuesday, October 22, 2013
Time Allowed: 2 hours, 7–9 pm
Instructions:

1. This is a closed-book, closed-notes examination.
2. There are **9** questions on **20** pages (including this 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 the class so far. *For example, no C++ classes, string class, nor pointers.*
6. For programming questions, you may *not* define additional helper functions or structures.

Student Name	Mr. Model Answer
Student ID	00000000
Email Address	nuts@comp2011.edu
Lecture & Lab Section	L123 LA123

For T.A.

Use Only

Problem	**Score**
1	/ 12
2	/ 8
3	/ 14
4	/ 12
5	/ 8
6	/ 11
7	/ 9
8	/ 14
9	/ 12
Total	/ 100

Problem 1 [12 points] True or false

Indicate whether the following statements are *true* or *false* by circling **T** or **F**. You get 1.5 point for each correct answer, −0.5 for each wrong answer, and 0.0 if you do not answer.

[✓ **T**] **F** (a) Assigning a double value to a char variable will NOT cause a compilation error.

T [✓ **F**] (b) In C++, a positive integer, when used in a boolean expression, is interpreted as true and a non-positive integer is interpreted as false.

[✓ **T**] **F** (c) A while loop can always be rewritten using a for loop and vice versa.

T [✓ **F**] (d) The following piece of code will cause an infinite loop, regardless of what statements we include within the loop.

```
int a = -5;
while (a)
{
        // statements
        // ...
}
```

[✓ **T**] **F** (e) Suppose 32-bit single precision is used to represent a floating point number. Increasing the number of exponent bits will make the maximum representable floating-point number bigger.

T [✓ **F**] (f) What is the *truth value* of the following boolean expression when D is 17, P is 3, and Q is 8? (Note that the order of operators precedence is: / is higher than ==, and ! is higher than + which is higher than >.)

```
    (P == D/Q)||(!P + Q > D)
```

T [✓ **F**] (g) The following piece of code causes a compilation error.

```
float scores[100];
cout ≪ scores[100];
```

T [✓ **F**] (h) A recursive function only allows argument(s) to be passed to its formal parameters by its/their values and not by its/their references.

Problem 2 [8 points] Loops

Re-write the following code fragment using a *do-while* loop so that it will give the same outputs.

```
int n;
cout << "Enter a non-negative integer:  ";
cin >> n;

while (n < 0)
{
    cout << "Invalid negative input." << endl;
    cout << "Enter a non-negative integer:  ";
    cin >> n;
}
```

Answer:

```
int n;

do
{
    cout << "Enter a non-negative integer:  ";
    cin >> n;

    if (n < 0)
        cout << " Invalid negative input." << endl;
} while (n < 0);
```

Grading scheme:

- 1 point for defining n.
- 2 points for the next 2 lines: cout and cin.
- 2 points for the if statement.
- 2 points for the do-while condition.
- 8 points only if everything is correct.
- Repeated minor syntax errors: each 0.5 point, max 1 point.

Problem 3 [14 points] Program Outputs

For each of the following program code segments, write down its expected output. Of course, you may assume that iostream header file is properly included.

(a) [4 points]

```
int k = 5;
int n = (100 % k ? k + 1 : k - 1);
cout << n << endl;
```

Answer: 4

(b) [4 points]

```
enum color {RED, ORANGE, YELLOW, GREEN, BLUE, VIOLET};
color shirt, pants;

shirt = RED;
pants = BLUE;
cout << shirt << " " << pants << endl;
```

Answer: 0 4

Grading scheme: 2 for each answer

(c) [6 points] Please write down the output if the user types in the following message, which consists of 50 characters (including spaces and the period at the end), and then presses 'Enter':

I never let schooling interfere with my education.

```
const int LENGTH = 12;
char message[LENGTH];
int i = 0;

cout << "Enter a sentence on the line below." << endl;
do
{
    cin >> message[i];
    ++i;
}
while ( (i < LENGTH - 1) && (message[i] != '\n') );

message[i] = '\0';
cout << message << endl;
```

Answer: Ineverletsc

Grading scheme: scan from left to right, 0.5 point for each correct letter.

Stop when an error occurs.
Give full mark if the whole output is all correct.

Problem 4 [12 points] Parameter Passing Methods in Function

For each of the following programs, write down its expected output.

(a) [6 points]

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

int increment(int &x) { return (x+1); }

int main(void)
{
    int a = 5;
    cout << increment(a) << endl;
    cout << a << endl;
    return 0;
}
```

Answer: 6
5

Grading scheme: 3 for each answer

(b) [6 points]

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

void modify_string(char x[ ], char y)
{
    y = '\0';
    x[8] = '3';
}

int main(void)
{
    char x[ ] = "Fall 2012!";          // there is a space between "Fall" and "2012"
    modify_string(x, x[4]);
    cout << x << endl;
    return 0;
}
```

Answer: Fall 2013!

Grading scheme: no partial credits.

Problem 5 [8 points] Break and Continue

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

int main(void)
{
    // s contains the 26 characters in alphabetical order
    char s[ ] = "abcdefghijklmnopqrstuvwxyz";

    for (int j = 0; j < 2; ++j)
    {
        for (int k = 0; k < 3; k += 2)
        {
            int n = j+k;
            if (n > 26)
            {
                cout << "Ooops!" << endl;
                return 1;
            }
            else if (n%3)
                cout << s[n];
            else
                continue;
        }
    }

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

(a) [4 points] Write down the output after running the above program.

Answer: cb

(b) [4 points] What is the output if the **continue** statement is replaced by the **break** statement?

Answer: b

Grading scheme: no partial credits.

Problem 6 [11 points] Crossword Puzzle as an Array of C Strings

column

row	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1			W				
2		F	E	E	L		
3		R			I		
4	G	O		H	O	M	E
5		M	E	A	N		

Figure shows a 5 × 7 crossword puzzle filled with words. Complete the following program to print out the various words or phrases in the puzzle. Use an asterisk '∗' to represent each unfilled square of the crossword puzzle in your 2D char array.

Grading scheme: Part (a)

- all correct 5 points.
- if only the 5x7 chars in s are correct, 2 points.
- if it fills up the remaining chars in the 5x10 char array, 4 points.
- must set s[3][7] = NULL character.
- (a) all correct 5 points.

Part (b): 3 points, no partial credits.
Part (c): total 3 points; 1 point for if only one part is correct.

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

int main(void)
{
    /* (a) Initialize the following 2D char array, s, to capture all the
     *      words in the above crossword puzzle.
     */
    char s[ ][10] = {                                    /* ANSWER HERE */
        "**W****",
        "*FEEL**",
        "*R**I**",
        "GO*HOME",
        "*MEAN**"
    };

    /* (b) In the space provided below, write an expression in term of the 2D char
     *      array, s, to print the phrase "GO*HOME" on the 4th row of the array.
     *      Note: NO ADDITIONAL "<<" operations are allowed in your answer.
     */
    cout <<                              /* ANSWER HERE */ s[3] << endl;

    /* (c) In the space provided below, write a for loop with ONE statement in
     *      its body to print the word "LION" on the 5th column of the array.
     *      Note:
     *      1. You MUST answer by filling in the for-loop with a SINGLE statement
     *         that makes use of the array s.
     *      2. You are not allowed to simply do something like:
     *                   cout << "LION" << endl;
     *      3. You also cannot use more than one cout statement in the loop.
     */
    for (                      /* ANSWER HERE */ int j = 1; j < 5; ++j)
    {
        cout <<                              /* ANSWER HERE */ s[j][4];
    }

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

Problem 7 [9 points] Recursion: Greatest Common Divisor

A function named `gcd` takes two positive integer arguments (i.e. greater than zero) and returns the greatest common divisor of those two integers. You can assume the input arguments are always correct.

For example,

- `gcd(28, 70)` returns 14,
- `gcd(40, 50)` returns 10,
- `gcd(256, 625)` returns 1, and
- `gcd(42, 6)` returns 6.

One efficient method to find the greatest common divisor among two integers is called the *Euclidean Algorithm*. A recursive C++ implementation of the algorithm is shown below:

```
int gcd(int a, int b)
{
    if (b == 0)
        return a;
    else
        return gcd(b, a % b);
}
```

Trace the execution of `gcd(18, 88)`. That is, give all the recursive calls of `gcd(18, 88)`. The first one is given to you already.

Answer:

$$\begin{aligned} \text{gcd}(18, 88) &\longrightarrow \text{gcd}(88, 18) \\ &\longrightarrow \text{gcd}(18\ , 16) \\ &\longrightarrow \text{gcd}(16\ , 2) \\ &\longrightarrow \text{gcd}(2\ , 0) \end{aligned}$$

Grading scheme: 3 points for each recursive call.

Problem 8 [14 points] 2D Arrays

Consider the following program.

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

const int ROWS = 7;
const int COLS = 7;

void flip_image(char image[ ][COLS]);
void bend_diagonal(char image[ ][COLS]);

void swap(char& a, char& b)
{
    char temp;

    temp = a;
    a = b;
    b = temp;
}

void print_image(char image[ ][COLS])
{
    for(int i=0; i<ROWS; i++)
    {
        for(int j=0; j<COLS; j++)
            cout << image[i][j] << " ";
        cout << endl;
    }

}


// [the program continues on the next page...]
```

```
int main(void)
{
    char image[ROWS][COLS] = {
        {'#', '0', '0', '0', '1', '1', '1'},
        {'0', '#', '0', '0', '0', '0', '1'},
        {'0', '0', '#', '0', '0', '0', '1'},
        {'0', '0', '0', '#', '0', '0', '0'},
        {'0', '0', '0', '0', '#', '0', '0'},
        {'0', '0', '0', '0', '0', '#', '0'},
        {'0', '0', '0', '0', '0', '0', '#'}
    };

    cout << "Original image:  " << endl;
    print_image(image);

    cout << endl << "Image flipped over the diagonal:  " << endl;
    flip_image(image);
    print_image(image);

    cout << endl << "Image with bent diagonal:  " << endl;
    flip_image(image);
    bend_diagonal(image);
    print_image(image);

    return 0;
}
```

The program has the following output (see next page).

```
Original image:
# 0 0 0 1 1 1
0 # 0 0 0 0 1
0 0 # 0 0 0 1
0 0 0 # 0 0 0
0 0 0 0 # 0 0
0 0 0 0 0 # 0
0 0 0 0 0 0 #

Image flipped over the diagonal:
# 0 0 0 0 0 0
0 # 0 0 0 0 0
0 0 # 0 0 0 0
0 0 0 # 0 0 0
1 0 0 0 # 0 0
1 0 0 0 0 # 0
1 1 1 0 0 0 #

Image with bent diagonal:
# 0 0 0 1 1 1
0 # 0 0 0 0 1
0 0 # 0 0 0 1
0 0 0 # 0 0 0
0 0 # 0 0 0 0
0 # 0 0 0 0 0
# 0 0 0 0 0 0
```

Write the functions `flip_image` and `bend_diagonal`, so that the program gives an output which is *identical* to the one shown above. The two functions have the following specifications:

- `flip_image` flips the 2D array over its diagonal in the way appearing in the sample output above. That is, the elements that lie above the diagonal must be swapped with those that are below the diagonal. The elements on the diagonal should not be changed.
- `bend_diagonal` must bend the diagonal in the way appearing in the sample output above.

Your functions **must** satisfy the following **requirements**:

- Both functions must use exactly one double-nested `for` loop. No other loop is allowed.
- Both functions must use the `swap` function only inside the double-nested `for` loop.
- You are not allowed to use the assignment operator for any element of the array. For instance, you cannot write `image[7][0] = '#';`

Answer:

```
void flip_image(char image[ ][COLS])
{
    for (int i = 0; i < ROWS; i++)
        for (int j = 0; j < COLS; j++)
            if (i < j)
                swap(image[i][j], image[j][i]);
}

void bend_diagonal(char image[ ][COLS])
{
    for (int i = 0; i < ROWS; i++)
        for (int j = 0; j < COLS; j++)
            if (i == j && i > (ROWS/2))
                swap(image[i][j], image[i][(COLS-1) - j]);
}
```

Grading scheme: for both parts

- 7 points for each part.
- 1 point for setting up the double nested for loop correctly.
- -2 points if the output has some minor additional figures. e.g., instead of flipping the 1's in part (a), they 1's are copied.
- Repeated minor syntax errors: each 0.5 point, max 1 point.

Problem 9 [12 points] Eliminate Duplicates in an Array

Write a function named `eliminate_duplicates` that takes an array of positive integers (which may not be sorted) and eliminates all the duplicate integers in the array.

The function should take two arguments: an integer array `a[ ]`, its number of elements `n`. The function should not return any value. On the other hand, if any duplicate integers are eliminated, then the function should change the value of `n` so that, on return, its new value gives the number of distinct integers in the array `a[ ]`.

Here is an example. Suppose the array `a[ ]` passed to the function is as shown below, and `n` is 11.

index	0	1	2	3	4	5	6	7	8	9	10
value	58	26	91	26	70	70	91	58	58	58	66

Then, after the function is called, the array should becomes the following:

index	0	1	2	3	4	5	6	7	8	9	10
value	58	26	91	70	66	??	??	??	??	??	??

and `n` should become 5. The question marks in the array elements after the 5th cell indicate that it does not matter what numbers are in those cells when the function returns.

Implement the function `eliminate_duplicates` described above in the skeleton code below by following the comments in the code.

Note: Only non-recursive implementation is allowed. Moreover, except the array `a[ ]`, no additional arrays are allowed.

Grading scheme: for both parts

- 6 points for each part.
- Part (a): 3 points partial credits for clearly showing how nested loops are set up that try to eliminate the duplicates.
- Part (b): 4 points for packing the distinct integers to the front; 2 points for correct value of n.
- Repeated minor syntax errors: each 0.5 point, max 1 point.

```
void eliminate_duplicates(int a[ ], int& n)
{
    // (a) Change all duplicates to -1. e.g., if a[ ] originally contains:
    // 58 26 91 26 70 70 91 58 58 58 66, it will become
    // 58 26 91 -1 70 -1 -1 -1 -1 -1 66

    for (int j = 0; j < n; ++j)
    {
        if (a[j] != -1)                     // Skip the duplicates that are already converted
        {
            for (int k = j+1; k < n; ++k)
                if (a[k] == a[j])                       // Convert the duplicates to -1
                    a[k] = -1;
        }
    }

    // (b) Pack distinct integers in a[ ]. So a[ ] will become:
    // 58 26 91 70 66 ?? ?? ?? ?? ?? ??, where ?? indicates that it does
    // not matter what numbers are in those cells.
    // When the function returns, n is also updated to 5 in this example.

    // Now move all distinct integers to the front
    int num_distinct_int = 0;                      // Count the number of distinct integers

    for (int i = 0; i < n; ++i)
    {
        if (a[i] != -1)                         // Get a distinct integer when it is not -1
        {
            a[num_distinct_int] = a[i];
            num_distinct_int++;
        }
    }

    n = num_distinct_int;
}
```