COMP 2011 Midterm - Fall 2014 - HKUST

Date: October 25, 2014 (Saturday)
Time Allowed: 2 hours, 2–4pm
Instructions:

1. This is a closed-book, closed-notes examination.
2. There are **9** questions on **18** 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 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	000
Email Address	
Lecture & Lab Section	

For T.A. Use Only

Problem	**Score**
1	/ 15
2	/ 6
3	/ 6
4	/ 8
5	/ 12
6	/ 12
7	/ 18
8	/ 11
9	/ 12
Total	/ 100

Problem 1 [15 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) The `enum` keyword is used to define a new data type that consists of a finite set of named integer variables.

T **F** (b) Any `switch` statement may be re-written as an `if-else-if` statement.

T **F** (c) The following piece of code will cause an infinite loop to run forever in practice, regardless of what statements we put within the loop or outside the loop.

```
// statements
// ..

while (true)
{
    // statements
    // ...
}
```

T **F** (d) What is the truth value of x after running the following code?

```
bool x = true;
int j = 1;
if (j >= 3)
if (j > 7)
    x = true;
else
    x = false;
```

T **F** (e) The index of the last element of an array with n elements is n.

T **F** (f) If function A wants to call function B, then function B must be defined before the definition of function A.

T **F** (g) Overloaded functions are functions with the same name but different return types.

T F (h) Given the following two functions

```
void f(int num);
int g(int num);
```

the following statement is illegal.

```
g(f(10));
```

T F (i) The following recursive function can be used to compute the factorial of any non-negative integer (that is, any integer that is greater than or equal to zero) correctly.

```
int factorial(int n)
{
    if (n < 0)
        return -1;
    else
        return n * factorial(n-1);
}
```

T F (j) Even the `main` function can be recursive. That is, the `main` function may also call itself. For example, the following program is syntactically correct and runs with no errors.

```
#include <iostream>
using namespace std;
int x = 3;

int main( )
{
    if (x <= 0)
        return -1;

    cout << --x << endl;
    main( );
    return 0;
}
```

Problem 2 [6 points] Arithmetic and Boolean Operators

Determine the output for each of the following program codes.

(a) [2 points]

```
int a = 10;
int b = 20;
cout << ++a+b-- << endl;
```

Answer: 31

(b) [2 points]

```
int x = 10;
int y = 20;
int& z = x;

if (y = z)
    cout << x << endl;
else
    cout << y << endl;
```

Answer: 10

(c) [2 points] Determine the truth value of the following boolean expression:

`false` || `true` && `false` || `true`

i. [1 point] if the `||` operator has a higher precedence than the `&&` operator.

Answer: true

ii. [1 point] if both of the `||` operator and the `&&` operator have the same precedence and they are right-associative.

Answer: true

Problem 3 [6 points] Switch and enum

What is the output of each of the following programs that make use of `enum` or `switch`?

(a) [3 points]

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

enum test { A, B = 32, C };

int main( )
{
    cout << A << " " << B << " " << C << endl;
    return 0;
}
```

Answer: 0 32 33

(b) [3 points]

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

int main( )
{
    int choice = 1;

    switch (choice)
    {
        case 1: cout << "HKUST" << endl;
        case 2: cout << "HKU" << endl;
        case 3: cout << "CUHK" << endl;
        default: cout << "Error" << endl;
    }

    return 0;
}
```

Answer:

```
HKUST
HKU
CUHK
Error
```

Problem 4 [8 points] Loops and Continue

(a) [4 points] What is the output of the following program?

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

int main( )
{
    int rows = 4, k = 1;
    for (int i = 1; i <= rows; ++i)
    {
        for (int j = 1; j <= i; ++j, ++k)
            cout << k << ' ';                    // Print one space after the value of k
        cout << endl;
    }
    return 0;
}
```

Answer:
```
1
2 3
4 5 6
7 8 9 10
```

(b) [4 points] Now we add a `continue` statement. What is the new output?

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

int main( )
{
    int rows = 4, k = 1;
    for (int i = 1; i <= rows; ++i)
    {
        for (int j = 1; j <= i; ++j, ++k)
        {
            if (k%2)
                cout << k << ' ';                // Print one space after the value of k
            else
                continue;
        }
        cout << endl;
    }
    return 0;
}
```

Answer:
```
1
3
5
7 9
```

Problem 5 [12 points] Check If An Array Is Sorted By An Iterative Method and A Recursive Method

(a) [5 points] Given an array of integers, check if the array elements are sorted in ascending order (i.e., from the smallest to the largest) using an *iterative* function.

Note: You may assume that size of the array is alway greater than zero.

```
/* Inputs:
 *      x: an array of const integers
 *   size: size of the const integer array x
 */
bool iterative_check_ascending(const int x[ ], int size)
{
    // ANSWER: ADD YOUR CODE HERE
    for (int j = 0; j < size-1; ++j)                          // 1 point
        if (x[j] > x[j+1])                                    // 2 point
            return false;

    return true;                                              // 2 point
}
```

(b) [7 points] Repeat part (a) but using a *recursive* function.

```
/* Inputs:
 *      x: an array of const integers
 *   size: size of the const integer array x
 */
bool recursive_check_ascending(const int x[ ], int size)
{
    // ANSWER: ADD YOUR CODE HERE
    if (size == 1)                                    // Base case: 2 points
        return true;

    if (x[size-2] > x[size-1])                        // false condition: 2 points
        return false;
    else
        return recursive_check_ascending(x, size-1);  // Recurive case: 3 points
}
```

Problem 6 [12 points] Parameter Passing

Determine the output of the following program which calls various functions using different parameter passing methods.

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

int A(int a, int& b)
{
    a += b;
    b += a;
    return a+b;
}

int B(int& a, int b)
{
    a += b;
    b += a;
    return a+b;
}

int C(int& a, int& b)
{
    a += b;
    b += a;
    return a+b;
}

int D(int& a, int b)
{
    a += b;
    b += a;
    return A(a,b)+b;
}
```

```
int main( )
{
    int x = 0, y = 1;
    cout << A(x,y) << endl;
    cout << B(x,y) << endl;
    cout << C(x,y) << endl;
    cout << x << ' ' << y << endl;                    // Print one space

    x = 0, y = 1;                                     // Reset the values of x and y
    cout << D(x,y) << endl;

    return 0;
}
```

Grading scheme:
2 points for each of function A, B and C outputs.
1 points for each of x and y on the 4th line.
4 points for function D output.

```
3
6
10
4 6
13
```

Answer: _____

Problem 7 [18 points] Sub-string in a Sentence

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

const int LENGTH = 100;
void to_lower_case(const char input[ ], char result[ ]);
void only_letters(const char input[ ], char result[ ]);
bool is_substring(const char s1[ ], const char s2[ ]);

int main(void)
{
    char input_sentence[LENGTH];        // May contain any keyboard char except '\n'
    char s[LENGTH];                     // Contains only English letters in lower case
    char input_in_lower_case[LENGTH];                  // input_sentence in lower case
    char input_with_only_letters[LENGTH];          // input with no non-English letters

    cout << "Enter a sentence:  ";
    cin.getline(input_sentence, LENGTH);
    to_lower_case(input_sentence, input_in_lower_case);
    only_letters(input_in_lower_case, input_with_only_letters);

    cout << "Enter a substring:  ";
    cin >> s;                   // Assume: s contains only English letters in lower case
    cout << boolalpha << is_substring(input_with_only_letters, s) << endl;
    return 0;
}
```

The program above first reads in a sentence using the function `cin.getline` until the user types in the newline character. Thus, the string, `input_sentence` consists of any character one may type, including capital letters, punctuations, whitespaces and other symbols.

Then a simple string `s` is read in using the `<<` operator of `cin` which is assumed to contain only some combination of the 26 English letters in lower case. Now we want to check if the string `s` is a substring of `input_sentence` using the boolean function `is_substring`. But before we do so, we will first

1. convert all alphabetical characters in the `input_sentence` to their lower case by calling the function `to_lower_case`, and
2. remove all non-alphabetical characters in the sentence by calling the function `only_letters`.

For example, if the input sentence is

A man, a plan, a canal: Panama.

After the first step of calling the function `to_lower_case`, it becomes

a man, a plan, a canal: panama.

Then after the second step of calling the function `only_letters`, it is turned into

amanaplanacanalpanama

And if the second input `s` is "ana" then the boolean function `is_substring` should return `true`; on the other hand, if `s` is "hkust", then it should return `false`.

Complete the definition of the 3 required functions on the following pages.

Remarks:

- You may assume that all inputs are valid and they consist of at most (LENGTH-1) characters.
- You are NOT allowed to use any string library function except the following:

  ```
  int strlen(const char str[ ]);
  ```

  which may be used to get the length of any string `str`.
- You may define additional helper function(s) if you find the need to do so. Make sure you indicate clearly where the helper functions will be put.

Grading scheme:
-0.5 for each different syntax error; max -2 points.

```
#include <cstring>

/* Part (a) [5 points]
 * Copy the input C-string to the result C-string but change any capital
 * letter in the input string to its lower case. Non-alphabetical characters
 * are copied with no modification.
 */
void to_lower_case(const char input[ ], char result[ ])
{
    // ANSWER: ADD YOUR CODE HERE
    for (int j = 0; j <= strlen(input); j++)          // 1 point: correct loop set up
    {                                                 // 1 point: the ending NULL CHAR
        if (input[j] >= 'A' && input[j] <= 'Z')       // 2 points: case change
            result[j] = input[j] - 'A' + 'a';
        else
            result[j] = input[j];                     // 1 point: copying the rest
    }
}


/* Part (b) [5 points]
 * Copy the input C-string to the result C-string but remove all
 * non-alphabetical characters in the input.
 */
void only_letters(const char input[ ], char result[ ])
{
    // ANSWER: ADD YOUR CODE HERE
    int k = 0;
    for (int j = 0; j < strlen(input); j++)           // 2 points: correct loop/var setup
    {
        if (input[j] >= 'a' && input[j] <= 'z')       // 2 points: correct filtering
            result[k++] = input[j];
    }

    result[k] = '\0';                                 // 1 point: the ending NULL CHAR
}
```

```
/* Part (c) [8 points]
 * Check if s2 is a substring of s1.
 * Assumption: Both s1 and s2 contain only alphabetical characters in lower case.
 */



/*
 * An additional helper function which is an overloaded function.
 * Check if s2 is a substring of s1 starting from its j-th character.
 * Assumption: Both s1 and s2 contain only alphabetical characters in lower case.
 */
bool is_substring(const char s1[ ], const char s2[ ], int j)
{
    for (int k = 0; k < strlen(s2); ++k, ++j)                    // 1 point: loop setup
    {
        if (s1[j] != s2[k])                                      // 2 points: Mismatch in any array items
            return false;
    }

    return true;                                                 // 1 point: if reach the end of the word s2
}



bool is_substring(const char s1[ ], const char s2[ ])
{
    // ANSWER: ADD YOUR CODE HERE
    for (int j = 0; j <= strlen(s1) - strlen(s2); ++j)           // 1 point: loop setup
    {
        if (is_substring(s1, s2, j))  // 3 points: the logic; need not be a helper function
            return true;
    }

    return false;
}
```

Problem 8 [11 points] About Recursive Function Calls

```
int fibonacci(int n)
{
    cout ≪ "Call fibonacci(" ≪ n ≪ ")"≪ endl;

    if (n == 0) return 0;
    if (n == 1) return 1;

    return fibonacci(n-1) + fibonacci(n-2);
}
```

The recursive function above finds the nth fibonacci number where n is a non-negative integer.

(a) [4 points] Write down the output of the above function, `int fibonacci(int n)`, when the function is called with $n = 3$.

```
Call fibonacci(3)
Call fibonacci(2)
Call fibonacci(1)
Call fibonacci(0)
Call fibonacci(1)
```

Answer: _____

Grading scheme:
0.8 point for each correct line;
-0.5 point for any extra output.

(b) [7 points] Write another recursive function,

```
int num_fibonacci_calls(int n);
```

which returns the total number of `fibonacci` function calls when it is called with the input n. Table 1 give some examples of the returned values of `num_fibonacci_calls(int)` for different values of n.

Note that (i) the function `num_fibonacci_calls` does not compute or print out the fibonacci number(s) but only tells you how many times the `fibonacci` function is called to produce the nth fibonacci number. (ii) Your answer must be a **recursive** function that works with the `main` function on the next page to produce the above results.

n	Returned Value
0	1
1	1
2	3
3	5
4	9
5	15
6	25

Table 1: Examples of values returned by num_fibonacci_calls(int).

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

int num_fibonacci_calls(int n)
{
    // ANSWER: ADD YOUR CODE HERE
    if (n == 0)                                    // 1 point: Base case #1
        return 1;

    if (n == 1)                                    // 1 point: Base case #2
        return 1;

    // 5 points: Recursive case
    return 1 + num_fibonacci_calls(n-1) + num_fibonacci_calls(n-2);
}




int main( )
{
    cout << "Enter a non-negative integer:  ";
    int n;
    cin >> n;
    cout << "no.  of fibonacci calls = " << num_fibonacci_calls(n) << endl;
    return 0;
}
```

Problem 9 [12 points] Tax Computation

Every Citizen in Loutacs (an imaginary country) needs to pay tax. The more you earn, the more tax you will pay.

There are two ways to compute one's tax in Loutacs.

First method: Progressive Rates

The rules for computing the tax using progressive rates are:

- No tax for the first $100 of the income
- 10% of the next $400 of the income
- 20% of the next $600 of the income
- 30% of the remaining income

Second method: Flat Rate

The Loutacs government will also calculate the tax according to the flat rate of 25% from the income.

The actual tax you pay is the smaller among the results computed by the above two methods.

The table below give some examples.

Citizen	Income	First $100	Next $400	Next $600	Remainder	Progressive Tax	Flat Rate Tax	Final Tax
A	75	75	0	0	0	0	18	0
B	120	100	20	0	0	2	30	2
C	800	100	400	300	0	100	200	100
D	9100	100	400	600	8000	2560	2275	2275

(a) [6 points] Write a function that takes the income as the input parameter and returns the tax amount computed using the Progressive Rates; truncate your result to an integer.

```
int progressive_rate_tax(int income)
{
    // ANSWER: ADD YOUR CODE HERE
    double tax = 0.0;

    // 1 point: case #1
    int chargeable_remaining = income - 100;
    if (chargeable_remaining <= 0)
        return 0;

    // 2 point: case #2
    tax += (chargeable_remaining <= 400) ? chargeable_remaining*0.1 : 400*0.1;
    chargeable_remaining -= 400;
    if (chargeable_remaining <= 0)
        return static_cast<int>(tax);

    // 2 point: case #3
    tax += (chargeable_remaining <= 600) ? chargeable_remaining*0.2 : 600*0.1;
    chargeable_remaining -= 600;
    if (chargeable_remaining <= 0)
        return static_cast<int>(tax);

    // 1 point: case #4
    tax += +chargeable_remaining*0.3;
    return static_cast<int>(tax);
}
```

(b) [2 points] Write a function that takes the income as the input parameter and returns the tax amount computed using the Flat Rate; truncate your result to an integer.

```
int flat_rate_tax(int income)
{
    // ANSWER: ADD YOUR CODE HERE
    return static_cast<int>(income*0.25);                    // 2 points
}
```

(c) [4 points] Write a void function that takes the income as the input parameter, and prints out the actual tax that is to be paid. It should print out which of the two methods is used to compute the actual tax. This function is expected to call the two functions defined in parts (a) and (b).

The output should look like this if the Flat Rate method gives the lower tax:

```
You have to pay \$200 tax.
It is computed using the Flat Rate.
```

On the other hand, if Progressive Rates method gives the lower tax, the output should look like:

```
You have to pay \$80 tax.
It is computed using the Progressive Rates.
```

If the two methods give the same result, you may pick any method.

```
void tax_payable(int income)
{
    // ANSWER: ADD YOUR CODE HERE
    int progressive_tax = progressive_rate_tax(income);          // 1 point
    int flat_tax = flat_rate_tax(income);                        // 1 point

    if (progressive_tax < flat_tax)                              // 1 point
    {
        cout << "You have to pay $" << progressive_tax << " tax." << endl
             << "It is computed using the Progressive Rates." << endl;
    }
    else                                                         // 1 point
    {
        cout << "You have to pay $" << flat_tax << " tax." << endl
             << "It is computed using the Flat Rate." << endl;
    }
}
```