COMP 2011 Midterm Exam - Fall 2018 - HKUST

Date: November 3, 2018 (Saturday)

Time Allowed: 2 hours, 2–4pm

Instructions:

1. This is a closed-book, closed-notes examination.
2. There are **7** questions on **xx** pages (including this cover page).
3. Write your answers in the space provided in black/blue ink. *NO pencil please, otherwise you are not allowed to appeal for any grading disagreements.*
4. All programming codes in your answers must be written in the ANSI C++ version as taught in the class.
5. For programming questions, you are **NOT** allowed to define additional helper functions or structures, nor global variables unless otherwise stated. You **cannot** use any library functions not mentioned, nor the `auto` keyword in defining identifiers in the questions.

Student Name	Marking Scheme
Student ID	
Email Address	
Lecture & Lab Section	

For T.A. Use Only

Problem	Score
1	/ 8
2	/ 10
3	/ 15
4	/ 15
5	/ 12
6	/ 16
7	/ 24
Total	/ 100

Problem 1 [8 points] Scope

Solution:

(a) [3 points]

1

Grading scheme: 3 points for correct answer

(b) [5 points]

11,99,12,99,13

Grading scheme: 1 point each number, maximum matching from left

Problem 2 [10 points] Loops

Solution:

(a) [5 points] 55777

(b) [5 points] 4455677788

Grading scheme: If the number of digits is correct: grade by correct digit of each location. Otherwise, grade by the longest string of correct digits.

(a) 1 point for each digit.

(b) 0.5 point for each digit.

Problem 3 [15 points] Function Parameter Passing

Solution:

Version#	Compile and run? YES/NO	Output if YES; One compilation error if NO
1	YES	8 4 8
2	NO	can't assign to x or y as they are const
3	YES	8 8 4
4	NO	can't assign to x or y as they are const
5	YES	8 8 4
6	NO	can't assign to x or y as they are const; can't bind const int reference to non-const reference return type

Grading scheme:
2.5 points per case.
YES/NO: 1 point
If YES: 0.5 point for each number.
If NO: 1.5 point for correct reason.

Problem 4 [15 points] Tower of Hanoi II

Solution:

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

// The statements must be in correct order to get the points
void toh2(int num_discs, char pegA, char pegB, char pegC) // 1 point
{
    if (num_discs <= 0) // Base case: 2 points
        return;

    if (num_discs == 1){ // Base case: 3 points
        cout << "move disc 1 alone from peg " << pegA << " to peg " << pegC << endl;
        return;
    }

    toh2(num_discs-2, pegA, pegC, pegB); // Recursion: 3 points

    // 3 points
    cout << "move discs " << num_discs-1 << " and " << num_discs
         << " together from peg " << pegA << " to peg " << pegC << endl;

    toh2(num_discs-2, pegB, pegA, pegC); // Recursion: 3 points
}

void toh2_v2(int num_discs, char pegA, char pegB, char pegC)
{
    if (num_discs == 1)    { // Base case
        cout << "move disc 1 alone from peg " << pegA << " to peg " << pegC << endl;
        return;
    }

    if (num_discs == 2)    { // Base case
        cout << "move disc 1 and 2 together from peg "
                << pegA << " to peg " << pegC << endl;
        return;
    }

    toh2_v2(num_discs-2, pegA, pegC, pegB);

    cout << "move discs " << num_discs-1 << " and " << num_discs
            << " together from peg " << pegA << " to peg " << pegC << endl;

    toh2_v2(num_discs-2, pegB, pegA, pegC);
}
```

Problem 5 [12 points] Lambda Expression

Complete the program below. There are 2 parts (a) and (b). Write you answer directly in the space provided.

Solution:

```
// [6 points]
[&]() { for (int j = 0; j < 10; ++j) sum_score += score[j]; }();
// 5 points:
// : 1 point: all capture by reference
// : 1 point: no formal parameter
// : 1 point: no calling argument
// : 2 points: correct for loop statement

// [6 points]
sum_score =
        [](int a[], int size)
        { int sum = 0; for (int j = 0; j < size; ++j) sum += a[j]; return sum; }
(score, 10);
// 7 points:
// : 1 point: no capture
// : 3 points: 2 correct formal parameters + correct calling arguments
// : 2 points: correct for loop statement
// : 1 point: correct return
```

Problem 6 [16 points] Ternary Search

Solution:

```
// recursive ternary search
int ternarySearch(const int arr[], int key, int first, int last){
    int middle1 = first + (last - first)/3; // 2 points
    int middle2 = first + (last - first)*2/3; // 2 points

    cout << "middle1 = " << middle1 << "\tmiddle2 = " << middle2 << endl;

    if (arr[middle1] == key) //base case 1: found, 2 points
        return middle1;
    if (arr[middle2] == key) //base case 2: found, 2 points
        return middle2;
    if (first > last) return -1; //base case 3: not found, 2 points
    // recursive cases
    if (key < arr[middle1]) // 2 points
        return ternarySearch(arr, key, first, middle1-1);
    if (arr[middle1] < key && key < arr[middle2]) // 2 points
        return ternarySearch(arr, key, middle1+1, middle2-1);
    if (arr[middle2] < key) // 2 points
        return ternarySearch(arr, key, middle2+1, last);
}
// any syntax error 0.5 point, same mistake deduct only once
```

Problem 7 [24 points] Array and Structure

(a) [6 points]

Solution:

```
// Part a
struct Char_Index
{
   char character;
   // 2 point for both correct data type and identifier name, otherwise 0
   // no partial marks for the 2 points
   int num_positions;
   // 2 point for both correct data type and identifier name, otherwise 0
   // no partial marks for the 2 points
   Position pos_list[MAX_NUM_POSITIONS];
   // 2 points for both correct data type, correct identier name and correct array size,
   // otherwise 0
   // no partial marks for the 2 points
};
```

(b) [6 points]

Solution:

```
// Part b
void add_char_position(Char_Index &index, int line, int pos)
{
   if (index.num_positions < MAX_NUM_POSITIONS)
   // check array full: 1.5 points
   {
      int& p = index.num_positions;
      index.pos_list[ p ].line = line;
      index.pos_list[ p++ ].pos = pos;
      // or any equivalent solutions
      // assign the line value : 1.5 point
      // assign the pos value: 1.5 point
      // increment the index.num_positions 1.5 point
   }
   // any syntax error: -0.5 points (note: same mistake across part (a) to part (c)
   //                                  should have the marks deducted once only.
}
```

(c) [12 points]

Solution:

```
// Part c
void build_indexes(const char story[][MAX_STR_LENGTH], int num_lines,
                   Char_Index index_arr[])
{
   /* initialize the indexes */
   for (int i = 0; i < NUM_CHARS ; i++) // 1 point
   {
      index_arr[i].character  = 'a' + i; // 2 point
      index_arr[i].num_positions = 0;  // 1 point
   }

   char c;
   for (int i = 0; i < num_lines; i++) // 1 point
   {
      for (int j = 0; story[i][j] != NULL_CHAR; j++) // 1 points
      {
         if ( ((story[i][j] >= 'a') && (story[i][j] <= 'z'))
           || ((story[i][j] >= 'A') && (story[i][j] <= 'Z')) )
         {
            c = story[i][j] < 'a' ? story[i][j] - 'A' + 'a' : story[i][j];
            add_char_position(index_arr[ c - 'a' ], i, j);
         }

         // handling lowercase alphabets: 3 points
         // which includes e.g. checking lowercase alphabet: 1 pt
         //                     get the corresponding index_arr element
         //                     call the add_char_position() correctly
         // handling uppercase alphabets: 3 points
         // which includes e.g. checking uppercase alphabet: 1 pt
         //                     get the corresponding index_arr element
         //                     call the add_char_position() correctly

      }
   }
}
```

-------------------- END OF PAPER --------------------