COMP 2011 Midterm Exam - Fall 2019 - HKUST

Date: November 2, 2019 (Saturday)
Time Allowed: 2 hours, 2–4pm
Instructions:

1. This is a closed-book, closed-notes examination.
2. There are **6** 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	Mr. Marking Scheme
Student ID	
Email Address	
Lecture & Lab Section	

For T.A. Use Only

Problem	Score
1	/ 12
2	/ 9
3	/ 18
4	/ 16
5	/ 25
6	/ 20
Total	/ 100

Problem 1 [12 points] C++ Controls

Grading scheme: 3 points for each number (matching from the start)

Answer: **5 4 2 7**

Problem 2 [9 points] Parameters Passing and Return Type

Grading scheme: 3 points for each blank

Answer:

```
int& fun(int x, int& y)
{
   x = 2 * x;
   y = y * x;
   return (y);
}
```

Problem 3 [18 points] Lambda Expressions

Grading scheme: 3 points for each sub-question. 1 point for each output value.

(a) **Answer: 1.5, -2, 1**

(b) **Answer: 1, -2, 1**

(c) **Answer: compilation error**

(d) **Answer: 1.5, -2, 1**

(e) **Answer: 1.5, 2, 1**

(f) **Answer: 1, 2, 1**

Problem 4 [16 points] Recursion: Same or Assimilation Game

Grading Scheme: Starting from the top, the longest sub-sequence (length ≥ 2) in your answer which matches with the solution will be located. 2 points for each output in the sub-sequence.

Solution:

```
(0, 2)
(0, 3)
(1, 2)
(1, 1)
(1, 0)
(2, 0)
(2, 1)
(3, 1)
```

Problem 5 [25 points] Array and Structure

Grading Scheme: -0.5 for syntax error(s) in a), -1 point for syntax error(s) in b), and -0.5 for syntax error(s) in c)

Solution:

```
/*  Initialize the super tic-tac-toe
 *  Grids in both physical board and super board are
 *  set to be EMPTY (not marked)
 *  Parameter: game - a struct which holds both physical and super board
 */

// syntax error if use 0 instead of EMPTY
void initGame(TicTacToe& game){
    for (int i=0; i<N; i++)
        for (int j=0; j<N; j++)
            setArray(game.physicalboard[i][j], EMPTY);    // 3 points
    setArray(game.superboard, EMPTY);    // 3 points
}

/*  Check whether there's a a horizontal, vertical,
 *  or diagonal line marked on a 3x3 board
 *  Parameter: array - a struct which holds a 3x3 board
 *  Return:    true - if found 3-in-a-line
 *             false - otherwise
 */
bool check3inline(const Array& array){
    // -0.5 point if forget to have array.grid[i][0] != EMPTY
    //check 3-in-a-row, 3 points
    for (int i=0; i<N; i++) {
        if (same(array.grid[i][0], array.grid[i][1], array.grid[i][2]) &&
                array.grid[i][0] != EMPTY)
            return true;
    }
    //check 3-in-a-column, 3 points
    for (int i=0; i<N; i++) {
        if (same(array.grid[0][i], array.grid[1][i], array.grid[2][i]) &&
                array.grid[0][i] != EMPTY)
            return true;
    }
    //check 3-in-a-diagonal, 3 points
    if (same(array.grid[0][0], array.grid[1][1], array.grid[2][2]) &&
            array.grid[0][0] != EMPTY)
        return true;
    if (same(array.grid[0][2], array.grid[1][1], array.grid[2][0]) &&
            array.grid[0][2] != EMPTY)
        return true;
```

```
    // can't find 3-in-a-line, 1 point
    return false;
    // Your code ends here
}

/* This function is used to visualize the game board
 * ' ' - empty, 'X' - player 1, 'O' - player 2
 *  Parameter:  game - a struct which holds both physical and super board
 */
void printBoards(const TicTacToe& game){
    const char symbol[3] = {' ', 'X', 'O'};
    cout << "Physical board:" << endl;
    cout<< "   0  1  2   3  4  5   6  7  8" << endl;
    cout << " |---------|---------|---------|" << endl;

    for(int i = 0; i < N * N; i++){    // 1 point
        cout << i << "|";    // 1 point
        for(int j = 0; j < N * N; j++){    // 1 point
            // 3 points
            cout << " "
                << symbol[game.physicalboard[i/N][j/N].grid[i%N][j%N]] << " ";
            if ((j + 1)%3 == 0) cout << "|";    // 1 point
        }
        cout << endl;    // 1 point
        if ((i + 1)%3 == 0)
            cout << " |---------|---------|---------|" << endl;    // 1 point
    }
}
```

or, alternatively,

```
//index i j can swap; a b can swap
for (int i = 0; i < N; i++)
    for (int a = 0; a < N; a++) {
        cout << i * N + a << "|";
        for (int j = 0; j < N; j++) {
            for (int b = 0; b < N; b++) {
                cout << symbol[game.physicalboard[i][j].grid[a][b]] << " ";
            }
            cout << "|";
        }
        cout << endl;
    }
cout << "|-----|------|------|" << endl;
}
```

Problem 6 [20 points] C String, Array and Recursion

(a) [2 points] 45778 is not a PPN. Convert it to a PPN according to the 2 rules given above.

Solution: 45432

Grading Scheme: -0.5 for each different syntax errors; -2 points max.

```
// (b) [4 points]
int string2int(const char s[], int x[])
{
    int k;
    for (k = 0; s[k] != '\0'; ++k)  // 1 point
        x[k] = s[k] - '0';          // 2 points for the correct conversion

    return k;                       // 1 point for correct return value
}


// (c) [7 points] 0 mark for non-recursive solution
bool is_pingpong(const int x[], int num_digits, int stepsize, int diff)
{
    if (num_digits <= stepsize) // Base case: 2 points
        return true;

    int difference = x[num_digits - 1] - x[num_digits - stepsize - 1];

    // 1 points for checking the absolute difference
    // 2 points for correct recursive call
    // 2 point for correct return value
    return ([](int a) { return (a > 0) ? a : -a; }(difference) == diff)
        && is_pingpong(x, num_digits - 1, stepsize, diff);
}


// (d) [7 points] 0 mark for non-recursive solution
void to_pingpong(int x[], int num_digits)
{
    if (is_pingpong(x, num_digits)) // Base case: 2 points
        return;

    to_pingpong(x, num_digits-1); // Recursion: 2.5 points

    // Fixing the next digit: 2.5 points
    int y = x[num_digits - 2] - 1;
    x[num_digits - 1] = (y < 0) ? x[num_digits - 2] + 1 : y;
}
```