# CSCI 103
## Fall 2013 Midterm

Name: ________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(20 points)</td>
</tr>
<tr>
<td>2</td>
<td>(5 points)</td>
</tr>
<tr>
<td>3</td>
<td>(6 points)</td>
</tr>
<tr>
<td>4</td>
<td>(10 points)</td>
</tr>
<tr>
<td>5</td>
<td>(15 points)</td>
</tr>
<tr>
<td>6</td>
<td>(15 points)</td>
</tr>
<tr>
<td>7</td>
<td>(14 points)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>(85 points)</strong></td>
</tr>
</tbody>
</table>


1. Short Answer/Multiple Choice (2 points each; 20 points total)
   a. A dynamically allocated variable, created with new, lives on the stack / the heap and is deallocated from memory when it goes out of scope / with the delete operator.

   b. What values evaluate to true in C/C++?

   c. What is the advantage to passing arguments to functions by reference?

   d. The expression a-- % b + ++c * d evaluates to what value, when a=8, b=2, c=3, and d=4?

   e. A function, f1, needs to take in an array of strings (C-style text strings) as an argument and return one of the strings in the array as the return value. The appropriate prototype for this function is:
      a) char *f1(char **);
      b) char f1(char *);
      c) char f1(char [][]);
      d) char *f1(char *

   f. Circle all statements that are true related to functions in C/C++
      a) Nested functions return in the reverse order as they are started/called (i.e. the latest called function is the first to finish)
      b) Local variables declared in a function are allocated on the stack area of memory belonging to the function
      c) Functions can return any number of values via the 'return' statement
      d) A function must return some value (it cannot return nothing)
g. In the code to the right, what will be printed:
   a) One
   b) Two
   c) Three
   d) Four

   ```cpp
   int nums[12] =
   {31, 28, 31, 30, 31, 31, 30, 31, 30, 31, 30, 31};
   int vals[4] = {1, 6, 10, 0};
   if( nums[vals[0]] > nums[vals[1]])
      cout << "One" << endl;
      cout << "Two" << endl;
   else if(nums[vals[0]] >= 30 || nums[0]==28)
      cout << "Three" << endl;
   else {
      cout << "Four" << endl;
   }
   ```

h. If an algorithm that operates on an input of size, n, requires $2n^2 + 1800n + 10000$ time steps to complete, it would be classified using big-O notation as:
   a) $O(n^2)$
   b) $O(n)$
   c) $O(10000)$
   d) $O(1)$

i. Given the line of code: `char* str = new char[30];` causes (how many) _______ bytes of memory to be allocated on the heap. The str variable is allocated on the (heap / stack).

j. `do { x -= 2; } while ( x > 5);` is equivalent to what loop
   a) while(x > 5){ x -= 2; }
   b) for( x = x-2; x > 5; x = x-2) { /* no body */ }
   c) while( --x > 5 ) { x = x - 1; }
   d) none of the above

2. Streams (5 points)
   Write a function called `divide` to prompt a user for two integers from the keyboard – call them $a$ and $b$. These two integers are to be used to carry out a division: $a / b$. Return the result of the division.
3. Examine the following code. Assume the program is run 3 times with the input from the user as shown below. Show what would be printed to the screen: (6 points)

```cpp
int main() {
    int x, y, z;
    cin >> x >> y >> z;

    if( x != y || y = z)
        cout << "Apple" << endl;

    else if( !(y < x) )
        cout << "Banana" << endl;

    if ( x - y - 2 )
        cout << "Carrot" << endl;
    else
        cout << "Dates" << endl;

    return 0;
}
```

4. Consider the function `replace()` below. It receives a number, `num`, an array of integers, `vals`, and the length of that array, `len`. It should return the `answer` array to the function which called `replace()`. (10 points)

   a) Complete the type declaration for the answer variable and the return statement.

   b) Given the following arguments indicate what `cout` will print and fill in the contents of the answer array with the values upon return.

   ```cpp
   num = 84; vals = {2, 3, 5, 7}; len = 4
   ```

   ```cpp
   int* replace(int num, int vals[], int len) {
       __________ answer = new int[len];

       for(int i=0; i < len; i++){
           answer[i] = 0;
           while( (num % vals[i]) == 0 ){
               num = num / vals[i];
               answer[i] += 1;
           }

           if(answer[i] == 0){
               answer[i] = vals[i];
           }
       }

       cout << num << endl;
       return __________;
   }
```

Fill in the values for the answer array below

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
5. Write a program to accept a number representing how many first names the user will enter from the command line (5 names maximum), and the actual first names, from the command line. Create an array dynamically that has the same size as number of names. Then store the names into the array. (15 points)

A sample command line is: ./names 3 tom dick harry

6. Examine the following function. Assume the 'first' and 'second' arrays have the contents shown below before the function starts executing and that len = 4. (15 points)
a) On the first line, complete the type of "dat" which should store the result of a dynamic array allocation of int's with two times 'len' number of elements, which you should also complete.
b) Fill in the contents of the 'dat' array shown below as the function executes.
c) Show what is printed to the screen

```c
int *m3(int first[], int second[], int len)
{
    int i, fptr=0, sptr=0;
    dat = ______________________;
    for(i=0; i < 2*len; i++){
        if(fptr == len || sptr == len)
            break;
        if(first[fptr] < second[sptr]){
            dat[i] = first[fptr++];
        }
        else {
            dat[i] = second[sptr++];
        }
    }
    while(fptr != len){
        cout << "Go" << endl;
        dat[i++] = first[fptr++];
    }
    while(sptr != len){
        cout << "SC" << endl;
        dat[i++] = second[sptr++];
    }
    return dat;
}
```

```plaintext
<table>
<thead>
<tr>
<th>first</th>
<th>2</th>
<th>3</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>second</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
```

Fill in 'dat' as the function executes:

```plaintext
<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Output to the screen:

7. Write the function reverseit that reverses the characters in a character array. You must also write main that calls reverseit. (14 points)

The prototype for reverseit is: void reverseit( char* inputString, char* outputString );

You are to prompt the user for the string to be reversed. Your program should read all the characters the user enters (which may include spaces and tabs). Pass the user’s string to the reverseit function. Output the reversed character array in main (not in reverseit).