ECE 150 Midterm

Fall 2022

Duration: 1 hour 30 minutes (90 minutes)

Instructions:

1. Do all the questions. The marks are shown next to each question.
2. If information appears to be missing from a question, make a reasonable assumption, state it, and proceed. The TAs or instructors will not answer any questions.
3. Closed books, closed notes, no use of calculators or computers, no use of videos, no use of compilers, and no use of any other resources besides your writing utensils.
4. There is space following each question to write your answer. There are extra empty pages at the end. Clearly indicate which questions you are using these pages for, if any.
5. No collaboration.
6. The declaration below must be signed to receive any marks.

Declaration

I confirm the following:

- I answered the questions on the midterm in their entirety on my own.
- I followed all the rules set by the midterm.

Signature:

1. For each error in the following C++ program, give a short reason for the error. No precise compiler error messages or fixes for the errors are requested. [2 marks]

```
#include <iostream>
main();

main() {
    std::cout << "Hello world!" <<
              << endl;
    return;
}
```

2. Given three integer variables x, y, and z, write a C++ program that moves the value of x to y, the value of y to z, and the value of z back to x. [2 marks]

3. Given the following two binary signed short numbers, perform the subtraction. Write the result of the subtraction in binary. [2 marks]

$$\begin{array}{r} 1100001011010011 \\ -\ \underline{0100001010100111} \end{array}$$

4. Complete this C++ program that counts the number of divisors of a given integer n. [4 marks]

```
#include <iostream>
int main();
int main() {
    int n{};

    std::cout << "Enter an integer: ";
    std::cin >> n;

    // Fill in your solution here:

    std::cout << "The number of divisors of " << n
              << " is " << d << std::endl;

    return 0;
}
```

5. Write the C++ function `ilog2` which returns the largest unsigned integer value *m* such that $2^m \leq n$ for the argument unsigned integer value *n*. This is the “integer logarithm base 2”. [6 marks]

```
unsigned int ilog2( unsigned int n ) {











}
```

6. A friend claims that, for a positive integer n, the following sum

$$1\times2\times3 + 2\times3\times4 + 3\times4\times5 + \cdots + n\times(n + 1)\times(n + 2)$$

can be calculated with the formula

$$\tfrac{1}{4}\times n\times(n + 1)\times(n + 2)\times(n + 3).$$

Write a C++ program that computes and outputs the values of both formulas for all integers between 3 and 13. [10 marks]

7. Write a C++ function even_sets_odd that takes an integer n and modifies its argument using passby-reference. [10 marks]

We number the bits of an integer starting with 0 at the least-significant (right-most) bit.

The function should do the following: starting with 0, we go through all even-numbered bits *i*.

- If bit *i* is 1, flip (turn a 0 into a 1 and vice versa) bit *i*+1.
- If bit *i* is 0, set bit *i*+1 to 1.

For example, for a 4-bit integer:

- Given `0b0001` the result will be `0b1011`.
- Given `0b1110` the result will be `0b0110`.

```
#include <iostream>
int main();

// Write your implementation here
```

8. Write a C++ program that queries the user for a number between 0 and 9 and then prints to the console a numeric pyramid (see examples below). [12 marks]

For input n = 0 print:
`0`
	For input n = 1 print:
	`␣1`
	`000`

For input n = 2 print:
`␣␣2`
`␣111`
`00000`
	For input n = 3 print:
	`␣␣␣3`
	`␣␣222`
	`␣11111`
	`0000000`

We use the symbol ␣ to highlight where to output a blank space.

9. An array X is a subarray of another array Y if the entries of X appear in the same order somewhere in Y. [12 marks]

Write three C++ functions:

```
int is_sub_array(int arr1[], int cap1, int arr2[], int cap2);
void remove_entries(int arr[], int cap, int start, int count);
void seek_n_remove(int arr1[], int cap1, int arr2[], int cap2);
```

These functions behave as follows:

1. The first function determines if `arr1` is a subarray of `arr2`. If `arr1` is a subarray of `arr2`, then the function returns the lowest index of `arr2` where `arr1` starts (so if the function returns n, then `arr1[0] == arr2[n]`, `arr1[1] == arr1[n + 1]`, `arr1[cap1 - 1] == arr2[n + cap1 - 1]`). If `arr1` is not a subarray of `arr2`, then the function returns `cap2`.
2. The second function eliminates these entries in `arr` going from `start` up to `start + count - 1`. The entry at index `start + count` is moved to index `start`, etc., and all subsequent entries in the array are also shifted by `count` indices. The last `count` elements of the array are filled with zeros.
3. The third function uses `is_sub_array` to determine whether `arr1` is a subarray of `arr2` or **vice versa**, and if so, the subarray is removed from the array containing it using `remove_entries`. You are guaranteed `cap1 != cap2`. If neither array is a subarray of the other, the function returns without any modification.

For example, take

```
int arr1[12] {1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6};
int arr2[3] {3, 4, 5};
seek_n_remove( arr1, 12, arr2, 3 );
// arr1 is now {1, 2, 6, 1, 2, 3, 4, 5, 6, 0, 0, 0} and
// arr2 is unchanged.
```