

Name _____ Login name _____

Section: _____ TA _____

General Instructions

- DO NOT PUT YOUR SSN ON ANYTHING!
- DO NOT WRITE YOUR NAME ON ANY PAGE EXCEPT THIS ONE!
- Turn off any noise making device, especially **CELL PHONES**. You may lose up to one letter grade if your device disturbs the peace of the exam.
- You have 50 minutes. Relax, **pace yourself**, and pay attention to the point values.
- The exam is 44 pts multiple choice, and pts programming and short answer.
- Do not add features that are not required by the problem. For example, if the instructions **don't say anything** about user input, then your program should **not** take user input. If you aren't sure, ask.
- Do problems you are confident about first. If you finish the problems you know, write what you do know about other problems to gain partial credit; but erroneous information may detract from that credit or irritate the grader, so don't make stuff up.
- Read *all* the directions *carefully* on each problem.
- Often writing a fast, rough version of a program in English or pseudocode will make your coding faster and more accurate. It also enables me to give partial credit in some circumstances.
- You may assume that input will not produce errors for the procedures described, unless the questions say otherwise.
- Do not do unnecessary testing. For example, testing for both $x < 0$ and $x \geq 0$ instead of using one test and then `else` would be considered unnecessary testing.
- Have fun!

Error definition

Errors do not always create problems in output, but should be considered errors anyway, as we do in class. This means errors include, but are not limited to:

- incorrect number of parameters
- writing to invalid memory
- reading from invalid memory
- other compilation errors
- other runtime errors

Errors do not include type coercion in arithmetic (for example, assigning a double to an int) unless that impairs the correctness of a program.

```
#include <iostream>

int myFind1(int key, int a[], int size);

int main(){
    int b[] = {2,3,4,5,9};
    std::cout << myFind1(5, b, 5) << std::endl;
    return 0;
}

int myFind1(int key, int a[], int size){
    int low = 0, middle, high = size;
    while (low < high) {
        middle = (low + high)/2;
        std::cout << low << " " << middle << " " << high << std::endl;
        if (key == a[middle]) {
            return middle;
        }
        else if (key < a[middle]) {
            high = middle - 1;
        }
        else {
            low = middle + 1;
        }
    }
    return -1;
}
```

1. (8 pts) Consider the program above, which compiles and runs. In the space below, write **exactly** what would be printed by the program on the lines provided. You may not need every line below, and you may add lines if you need them.

Note that the program has two print statements. Be careful about the order of printing.

2. (14 pts) Consider the following program, which may or may not compile and run. For each of the **seven** blanks, write what the program prints when the program runs correctly. If the line of code next to the blank contains an error¹, write “error”.

```
#include <iostream>

using namespace std;

int main(){

    double data[10] = {8,9,10};
    int x = 10;
    int *p = new int;
    int * xPtr = &x;

    cout << xPtr[0] << endl; _____

    cout << *xPtr[0] << endl; _____

    cout << *xPtr << endl; _____

    cout << data[3] << endl; _____

    cout << *data << endl; _____

    cout << *(data + 2) << endl; _____

    p = xPtr;

    cout << *p << endl; _____
    return 0;
}
```

¹see exam instructions for a definition of “error”.

3. (15 pts) Write an entire program. Using a loop, ask a user for an integer five times. Store the data in a single integer variable, summing as you go, and then calculate the floating-point average using a static type cast. Display the average with an appropriate message.

10. (10 pts) Write a function that uses tail recursion to raise a base integer to a positive integer exponent.

11. (10 pts) Why is the tail recursive version generally preferred? How does it behave differently? Be specific.

Use the code shown for the following two questions.

```
cout << "Enter a number of words: " << endl;  
cin >> size;
```

12. (5 pts) Declare an array that will hold the number of words specified above.

```
...  
cout << "Enter a string: ";  
cin >> temp;
```

13. (5 pts) Put the first word in the array, giving it only the space it requires.