Name: Tai Ngoc Bui
CSCI 3731
C++ - Homework 3

1. What is wrong with the following code and how would you fix it?

```
int min;
        for (int i=0; i<10; ++i) {
        int value;
        scanf("%d", &value);
        if(i=0) min = value;
        else if(value<min) min = value;
} // end of loop over input
printf("The smallest or the 10 values you entered was %d\n", min);
```

In the fifth line, within the if statement, variable i is assigned to value 0 instead of being compared with 0. As the condition is always false, the min variable will not be assigned to value in the fifth line. As a result, in the next line, value variable will be compared with some arbitrary garbage in the memory.

The fix would be to set if (i==0) min = value;

2. What is wrong with the following code and how would you fix it?

```
int* pointerToMax(int a, int b) {
        if(a>b) return &a;
        else return &b;
}
```

When passing variables a and b into a function, we merely pass a copy of those variables. Thus, any changes made to the parameters a and b inside the function are made only to the copy of a and b, not to the original value passed. The function will have absolutely no effect on the caller's original passed variables.

To fix the problem, we can alter the parameters into pointers

```
int* pointerToMax(int* a, int* b){
        if(*a > *b) return a;
        else return b;
}
```

3. What is wrong with the following code and how would you fix it?

```
int* ptr = NULL;
scanf("%d", ptr);
printf("You entered %d", &ptr)
```

The pointer is initialized to NULL with memory location 0x00. When you use scanf passing address 0x00, the result will be segmentation fault.

To fix the problem, we alter the code

Name: Tai Ngoc Bui
CSCI 3731
C++ - Homework 3

```
int ptr;
scanf("%d", &ptr);
printf("You entered %d\n", ptr);
```

4. Describe what each of the following declare:
(a) int* a; // declare a pointer pointing to an integer value
(b) const int b; // declare an int variable in which data cannot be modified
(c) const int* c; // declare a pointer which points to a const interger variable.
(d) int* const d; // a const pointer to a non-const integer variable.
(e) const int* const e; // declare a const pointer pointing to a const interger

5. What is the difference between the * and & operators?

& returns address of the variable, while the * does the opposite by dereferences the pointer and return the contents of the memory location it points to.

6. What is the difference between const int* a and int* const b?

const int* a: This is a pointer to const int; we can change the pointer but not the variable it points to.

int* const b: This is a const pointer to a non-const int variable. We can modify the data it points to but not the pointer itself.

7. Write the following two programs. Put each in a subdirectory under your hw03 directory.
(a) Write a program that prints the number of times you have run it. The first time you run it, it should print "1". The next time it should print "2", etc. Hint: how can you store information when the program is not running?
(b) Write a function that swaps the values of two variables. Write a program that calls this function, taking input from stdin and writing the values to stdout. Don't print from inside the function. Put the function in a separate file from your main. Hint: use pointers.