Part A: Due February 2

1. Explain, in your own words, the differences between passing an integer, a pointer to an integer, and a reference to an integer as parameters to a function. How are each used inside the function [2 marks] ? How can each affect what is outside of the function [2 marks] ?

Passing an integer:
It just copies the value to the function.
It does not change the value of the outside of the function, this kinds of passing is unilateral.

A pointer to an integer:
It can be changed to refer to another object, if you reference is initialized to an object.
It likes direct to use the value from the variable.
It can change the value of the outside of the function.

A reference to an integer:
It cannot have NULL references. And it cannot be changed to refer to another object, if you reference is initialized to an object. Actually, it just like a nickname of the integer to attached to that memory location.
It can change the value of the outside of the function.

2. Suppose you have a statically allocated array that contains space for 50 chars, but you currently have fewer than 50 chars to store. You would like to accommodate the possibility of adding or removing chars in the future (to a minimum of 1 and maximum of 50). Explain, in your own words, how to keep track of which entries of the array correspond to the data you are storing [3 marks] , and how to add and remove data [3 marks] . Note: at no point should it be necessary to allocate/free memory.

Arrays of chars have a special convention: An array of chars will be ended with a null char to indicate its end. Hence, we can use a "if- statement" and a loop to keep track of which entries of the array correspond to the data I am storing.

```
While(a[i]!='\0'){
    i++;
}
```

Or, we can use a counter to keep track of which entries of the array correspond to the data I am storing.

For add data, we could create a new array, and the size of the array should enough to

store data, and copy the old array, add the new elements.

For remove data: we could create a new array, and just copy the elements that we need to retain.

3. For each of the following problems, decide if an array would be a useful part of an efficient solution. Explain.
(a) Reading a sequence of numbers entered by the user and, when the user is finished, reporting the largest number entered [2 marks] .
(b) Reading a sequence of numbers entered by the user and, when the user is finished, reporting the average of all of the numbers entered [2 marks] .
(c) Reading a sequence of numbers entered by the user and, when the user is finished, reporting the numbers back in sorted order [2 marks] .

(a) Array is a useful part of an efficient solution, because we need use array to sort the number, and find the largest number to reporting, Array is easier to searching for a specific value.

(b) Array is a useful part of an efficient solution, because we need store data and, using loop and array to sum numbers and get the average of all of the numbers to reporting, and array is good at computer statistics.

(c) Array is a useful part of an efficient solution, because array is good at sort.

4. Consider the difference between statically and dynamically allocated memory. How can you tell in code that a value is being stored on the stack [1 mark] ? When does the memory for a variable on the stack get deleted [1 mark] ? Similarly, how can you tell in code that a value is being stored on the heap [1 mark] ? When does memory for a value on the heap get deleted [1 mark] ?

Stack is using the "last in, first out" way to stored on the stack. It means that the most recently produced codes will store on the top of the stack, and the most recently codes are stored as out first. When the variable is used (out), the stack will delete it atomically.

Heap has no organization compare with the stack, it can be stored anywhere in memory, and it would not be deleted by heap atomically. The programmers need delete it by them themselves, or until the function end system will delete it.

5.Three variables have been declared as follows: int a; , int *b; , and int *c; . The following statements are executed, in order:

1. a = 5;
2. b = &a;
3. c = b;
4. *c = 10;
5. a = 3;

Draw a diagram (using boxes and arrows, as in class) that illustrates what values are in a, b, and c after **each** of these lines is executed [5 marks, 1 mark each]. If one of these values is garbage/unknown, indicate so.