Lab Sheet 8

Pointers

1. Run the following program and understand the concept of pointers:

```
#include<stdio.h>
main()
{
        int a=10;
        int * b; b= &a; printf(" \n a=%d\t b=%d\t
        *b=%d\n",a,&a,*b);
}
```

2. Write a program which contains one regular integer variable **x** and one pointer to integer **ip.** Try out the following basic uses of pointers:
   - Assign **ip** the address of **x**.
   - Using the pointer as the argument to **scanf()**, read a value into the location **ip** points at (**i.e. x**) and then print out the value of **x.**
   - Assign a new value to ***ip**, and print out **x**.
   - Assign a value to **x** and print out ***ip**.

3. Write and test the following function: **void CharSwap(char *cp1, char *cp2);**
   The function must swap the characters in the memory locations stored by its two pointer arguments.

4. Write the following function:
   void input(int *small, int *medium, int *large);
   The function must read in 3 integers from the user, assign the smallest of the three to ***small**, the middle one to ***medium**, and the largest to ***large.** Write a main program which tests the function.

5. Write a function which finds both the minimum and maximum elements in an array of **doubles.** The function will need four arguments: the array, the length of the array, and two pointers to **double** through which the function can return the minimum and maximum values. Here is the prototype for the function:
   void MinMax(double *data, int length, double *retmin, double *retmax); Write a main( ) to test the function.

6. Declare an array of 5 integers. Print out the address of each element of the array. To print the address you can use **%p printf()** format which usually prints pointers in hexadecimal. Use the **%u (unsigned) printf()** format if you would rather not deal with

hexadecimal notation. Notice how the difference between each successive address is **sizeof(int)**.

7. Create an array of characters and fill it with a string. Define a character pointer and assign it the base address of the array. Use **pointer arithmetic to traverse the array and print out the characters one at a time**.

8. Write **a function which takes a char* as it's only argument and prints out each character in the array one at a time**. Use pointer arithmetic to step through the array. Recall that the pointer is passed by value so it can be modified without affecting the actual parameter.

9. Declare an array of **int** and a pointer to **int.** Assign the pointer the base address of the array. Print out the address of the array using **printf()**, increment the pointer using **++** and print out the address again. The value will have increased by **sizeof(int).**
   Try the same exercise with other types of arrays such as **char, float, double,** and **long double.** Notice how the value of the pointer is automatically increased by the appropriate amount.

10. It is the programmer's responsibility to keep a pointer into an array inside the array bounds. Compilers do not catch the error if the pointer is moved outside the array. Using an invalid pointer may write into memory being used for other variables. To demonstrate this, try the following. Define the variables:
    Char s1[5] = "AAA", s2[5], *s;
    Make **s** point just beyond the end of **s2** and write into the location **s** points at. Print out **s1.**

11. Write your own version of the library function **strcat().** strcat() takes two arguments, dest and source. It copies source onto dest**.** Use pointer arithmetic to manipulate the arrays. Declare the function like this: **void StringCopy(char *dest, const char *source);**

12. Write your own version of the library function **strcat().** Recall that **strcat()** takes two arguments, **dest and src**, and appends **src** onto the end of **dest**. **strcat()** must first find the end of the **dest** string, and then copy **src** onto the end of **dest.**
    Declare the function like this:
    void StringCat(char *dest, char *src);

13. Write a program to read and print an array of n numbers, then find out the smallest number. Also print the position of the smallest number.
    Use the following functions in the program.

void read_array(int *array, int n); void
print_array(int *array , int n);
void find_small(int *array, int n, int *small, int *pos);

14. Write a program to **read and display a matrix using pointers**.

15. Write a program to **read and display a 3D array using pointers**.

16. Write a program to **read and display values of an integer array**. Allocate space dynamically for the array.