Lab Exercise - Week 7

Experiment 1.
Program name: vowels.c

A string is an array of characters, where the last character is a special null character '\0'. Unlike ordinary arrays, you can scan and print an entire string using one scanf or printf call - using the format specifier "%s". For both the printf and scanf calls the variable is the name of the character array or equivalently the pointer to the first location in the array. The scanf command, by default, only reads the string till the first blank space and automatically appends the null character while storing it.

```
/* Declare a character array of size 30 */
char second_name[30];

/* You can declare and initialize a character array together.
Then the size of the array is calculated by C.
In the case below, it is 5 and not 4  because of the null character */
char first_name = "Adam" ;

scanf("%s", second_name);   /* To read a string input */
printf("%s %s\n", first_name, second_name); /* To print two strings */
```

First guess what will be the output of the next program. Then compile and execute it to see if your guess was correct

```
#include <stdio.h>

int main()
{
  char str[30];
  char *p;

  p = str;

  printf("Enter a word: ");
  scanf("%s", str);

  while (*p != '\0')
  {
    printf("%p: %s\n", p, p);
    p = p + 1;
  }

  return(0);
}
```

Write a C program (vowels.c) to read your first name and store it in a character array (string) and then do the following: (Do not use array indexing for this program. Use pointers instead.)

Count the number of vowels in your first name.
Store your name in another character array after removing all the vowels and print it.
Store your name in a third array after reversing the order of vowels in the name and print it.
Remember that C automatically appends a null character ('\0') to the end of every string. You can use that to terminate your while loops.

Experiment 2.
Program name: add_array.c

```
#include <stdio.h>

int main()
{
  int a[5], b[5], c[5];
  int counter = 0;

  printf("Enter 5 numbers in the first array\n");
  while(counter < 5)
  {
    scanf("%d",&a[counter]);
    counter = counter + 1;
  }
  counter = 0;
  printf("Enter 5 numbers in the second array\n");
  while(counter < 5)
  {
    scanf("%d",&b[counter]);
    counter = counter + 1;
  }
```

Complete the above program to fill the array c with term-by-term sum of arrays a and b and print the values in the array c.

Experiment 3.
Program name: peaks.c

Write a C program that takes a number n (the number of values), followed by a set of n values (integers) and display the number of "peaks" in the data. A value is a peak if it is strictly greater than its left neighbour and right neighbour.

For example, if the user inputs the following ten numbers
1, 5, 4, 6, 7, 8, 2, 9, 10, 11
Then there are two peaks (5 and 8).

Experiment 4.
Program name: last_count.c

Write a C program that takes 10 numbers and counts the number of times the last number is repeated. Also, print the number of entries smaller in value than the last number.

Experiment 5.

Program name: fibonacci_array.c

Write a C program to populate an array with the first 20 Fibonacci numbers. Print this array in the reverse order.

Experiment 6.
Program name: arrays.c

```
#include <stdio.h>

int main()
{
  int a[5];
  int i;

  i = 0;
  while (i < 5)
  {
    printf("%p: %d \t %p: %d\n", (a+i), *(a+i), &a[i], a[i]);
    i = i + 1;
  }

  printf("Enter 4 numbers\n");

  i = 0;
  while (i < 4)
  {
    scanf(" %d", &a[i]);
    i = i + 1;
  }
  /* The 5th element in the array will be garbage */

  i = 0;
  while (i < 5)
  {
    printf("%p: %d \t %p: %d\n", (a+i), *(a+i), &a[i], a[i]);
    i = i + 1;
  }

  return(0);
}
```

Run the above program and from the output try to guess the amount of memory allocated by gcc to store one integer variable.

Modify the above program by replacing the integer array with an array of characters, floats, and double precision floats. That is, change the "int" in the declaration of the array a to "char", "float" and "double".

Experiment 7 .
Program name: reverse.c

```
#include <stdio.h>

int main()
{
  int a[5] ={1, 1, 2, 3, 5};
  int *p, *q;

  p = &a[0];
  q = &a[4];

  while (p <= q)
  {
     printf("%d\n", *p);
     p = p + 1;
  }

  return(0);
}
```

Understand the above program and modify it to print the array a in the reverse order without introducing any more variables.
