Experiment No. 6

Pointers and Dynamic Memory Allocation

1. **Objective:** This experiment is intended to verify the concepts of pointers and dynamic memory allocation concepts in the C programming language. Overall, the example problems include pointer basics, pointer with arrays, pointer with strings, and dynamic memory allocation.

2. **Theoretical Background:** A pointer is a variable that holds the memory address of another object. For example, if a variable called p contains the address of another variable called q, then p is said to point to q. Therefore if q is at location 100 in memory, then p would have the value 100. To declare a pointer variable, use this general form:

```
type *var-name;
```

   Here, the type is the base type of the pointer. The base type specifies the type of object that the pointer can point to. Notice that the variable name is preceded by an asterisk. This tells the computer that a pointer variable is being created. For example, the following statement creates a pointer·to an integer:

```
int *p;
```

   C contains two special pointer operators: ***** and **&**. The **&** operator returns the **address of** the variable it precedes. The ***** operator returns the **value stored at the address** that it precedes. The ***** pointer operator has no relationship to the multiplication operator, which uses the same symbol.

   Pointers are required in -
   a. Dynamic memory allocation (accessing heap memory section)
   b. Function call by reference
   c. File accessing
   d. Accessing peripherals like monitors, printers, etc.

   Dynamic Memory Allocation can be defined as a procedure in which the size of a data structure (like Array) is changed during the runtime. C provides some functions to achieve these tasks. There are 4 library functions provided by C defined under <stdlib.h> header file to facilitate dynamic memory allocation in C programming. They are:

   - `malloc()`
   - `calloc()`
   - `free()`
   - `realloc()`

The “malloc” or “memory allocation” method in C is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form.

It doesn’t Initialize memory at execution time so that it initializes each block with the default garbage value initially.
Syntax:

```
ptr = (cast-type*) malloc(byte-size)
```

For Example:

```
ptr = (int*) malloc(100 * sizeof(int));
```

Since the size of int is 4 bytes, this statement will allocate 400 bytes of memory. And, the pointer ptr holds the address of the first byte in the allocated memory.

3. Software requirements:

- Any IDE that supports a C compiler (preferably Code::Blocks; any other IDE or text editor with a C compiler installed will also be accepted.)

4. Example Problems:

a. Write a program to find the smallest among three numbers using pointers.

Code:

```
#include<stdio.h>

int main(){

    int a,b,c,*pa,*pb,*pc;
    pa=&a;
    pb=&b;
    pc=&c;

    printf("Enter three integers: ");
    scanf("%d%d%d",pa,pb,pc);

    if (*pa<*pb && *pa<*pc)
        printf("the smallest number is %d",*pa);
    else if (*pb<*pa && *pb<*pc)
        printf("the smallest number is %d",*pb);
    else
        printf("the smallest number is %d",*pc);
    return 0;
}
```

b. Write a program to find the sum of all the elements of an array using pointers

Code:

```
#include<stdio.h>
int main(){
    int a[100],n,sum=0;
    printf("Enter the number of elements of the array: ");
    scanf("%d",&n);

    printf("Enter the array elements");
    for(int i=0;i<n;i++){
        scanf("%d",(a+i));
    }

    for(int i=0;i<n;i++){
        sum=sum+*(a+i);
    }

    printf("The sum of all the elements is = %d",sum);
    return 0;
}
```

c. Write a function to accept a string and count the number of vowels present in the string [use pointers].

Code:

```
#include<stdio.h>

int vowelCount (char *p){

    int counter = 0,i=0;

    while(*(p+i)){

         if(*(p+i) == 'A' || *(p+i) == 'E' || *(p+i) == 'I'
|| *(p+i) == 'O' || *(p+i) == 'U' || *(p+i) == 'a' || *(p+i)
== 'e' || *(p+i) == 'i' || *(p+i) == 'o' || *(p+i) == 'u'){

            counter++;

        }

        i++;
    }

    return counter;
}

int main(){

    char str[100];
    int n;
    printf("Enter a string: ");
    scanf("%s",str);

    n = vowelCount(str);

    printf("The number of vowels in the string is %d",n);

    return 0;
}
```

d. Write a program to remove all the empty spaces from a sentence. For example, if the input string is “The way to get started is to quit talking and begin doing”, the output should be “Thewaytogetstartedistoquittalkingandbegindoing”. [Store the input string in the dynamic memory].

Code:

```
#include<stdio.h>
#include<stdlib.h>

int main(){

    char *p,*q;
    int i=0,j=0;
    p = (char *)malloc(100*sizeof(char));

    printf("Enter a sentence: ");
    gets(p);

    q = (char *)malloc(100*sizeof(char));

    for(i=0;*(p+i);i++){

        if (*(p+i)!=' '){
            *(q+j) = *(p+i);
            j++;
        }

    }
    *(q+j) = '\0';

    printf("the output string is = %s",q);

    free(q);
    free(p);
```

```
        return 0;
    }
```

5. **Results after running the example programs:**
   Screenshots of the results:

6. **Lab report assignments (submit source code and screenshots of results):**
   a. Write a program to concatenate two strings using pointers.
   b. Write a function to copy one array to another by using pointers.
   c. Write a function to find out the dot multiplication of two vectors.

7. **Comment/Discussion on the obtained results and discrepancies (if any).**