CCC 204 Data Structures and Algorithms
LABORATORY REPORT :
LAB 4# - Pointers

Samano, Rajel Johann
Information Technology Program, CABECS
Colegio San Agustin – Bacolod

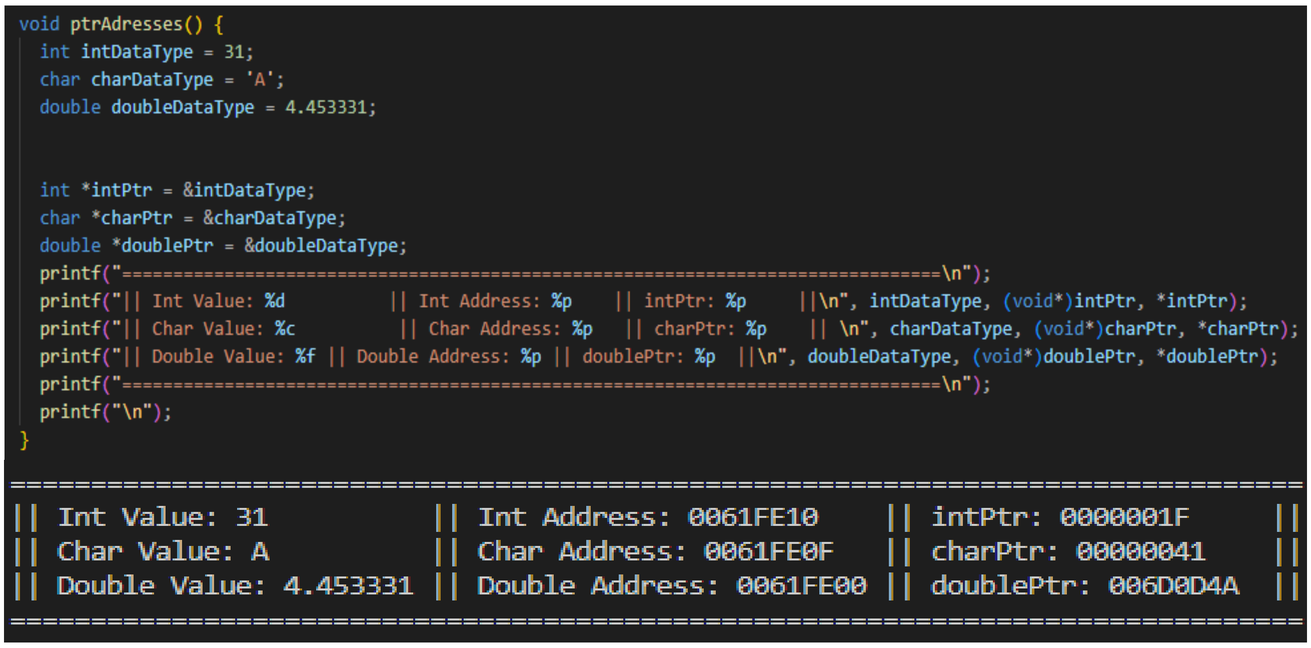
I. INTRODUCTION

For Laboratory Activity Number four is to follow the objectives of the exercises and answer questions. The Three main objectives as follows:

- Define, initialize and use pointers.
- Use pointers in pass by reference methods.
- Develop programs that use the main topic or concept introduced in the exercise.

II. IMPLEMENTATION / APPROACH

Figure 1. LA4_codeTasks.c with output

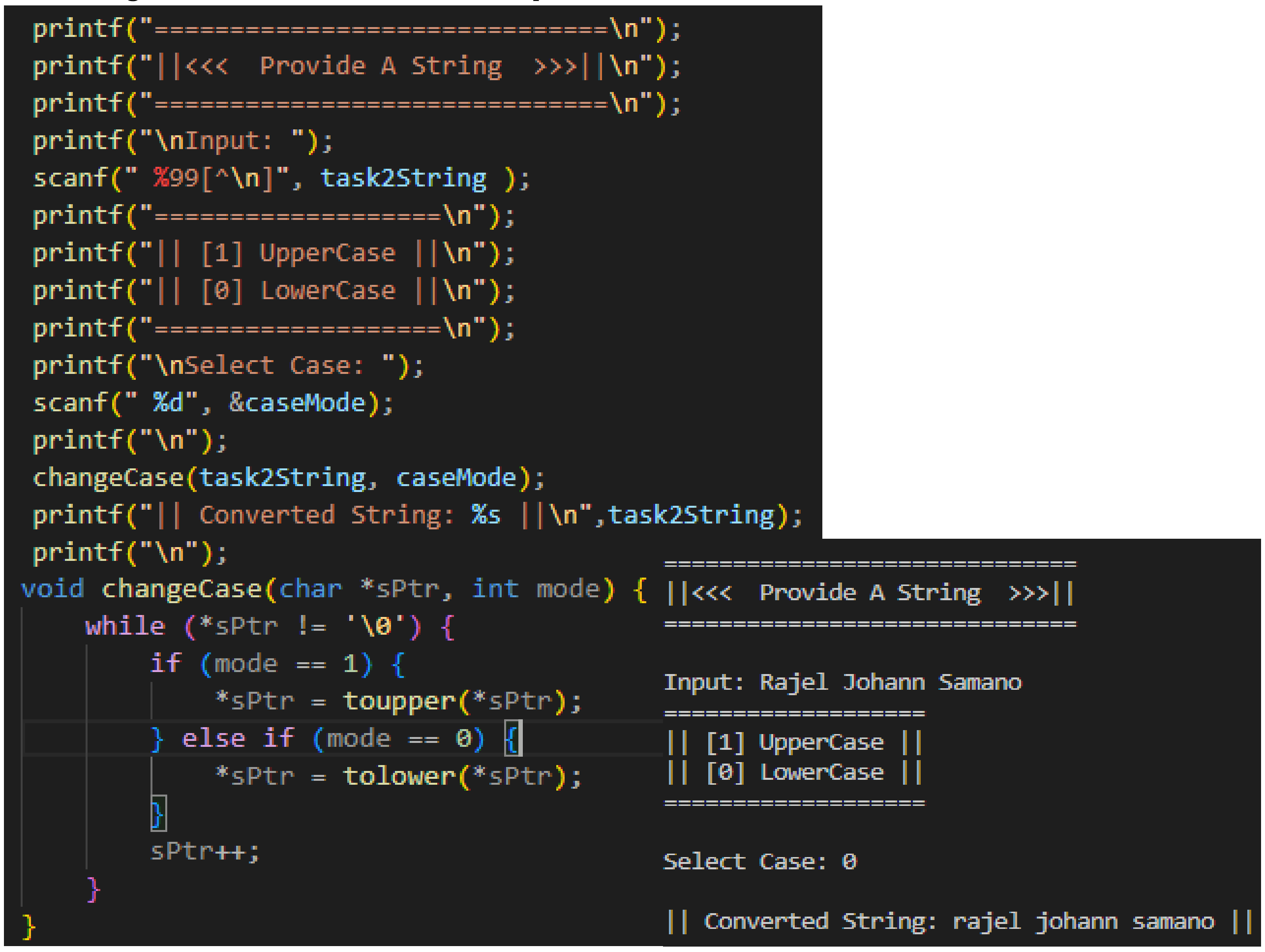


```c
void ptrAdresses() {
  int intDataType = 31;
  char charDataType = 'A';
  double doubleDataType = 4.453331;


  int *intPtr = &intDataType;
  char *charPtr = &charDataType;
  double *doublePtr = &doubleDataType;
  printf("================================================================================\n");
  printf("|| Int Value: %d          || Int Address: %p    || intPtr: %p     ||\n", intDataType, (void*)intPtr, *intPtr);
  printf("|| Char Value: %c          || Char Address: %p   || charPtr: %p    || \n", charDataType, (void*)charPtr, *charPtr);
  printf("|| Double Value: %f || Double Address: %p || doublePtr: %p  ||\n", doubleDataType, (void*)doublePtr, *doublePtr);
  printf("================================================================================\n");
  printf("\n");
}
```

```
================================================================================
|| Int Value: 31          || Int Address: 0061FE10    || intPtr: 0000001F     ||
|| Char Value: A          || Char Address: 0061FE0F   || charPtr: 00000041    ||
|| Double Value: 4.453331 || Double Address: 0061FE00 || doublePtr: 006D0D4A  ||
================================================================================
```

The task was to "**Print out the addresses and values of int, char, double data types as well as their respective pointers. Initialize the variables to any value you like**." My approach towards the task was to start by making variables for int, char, and double and then assigning them some values. After I have finished declaring the variables I then declared pointers for each data type after that I typed in printf to make the output for the given task.

Figure 2. LA4_codeTasks.c with output

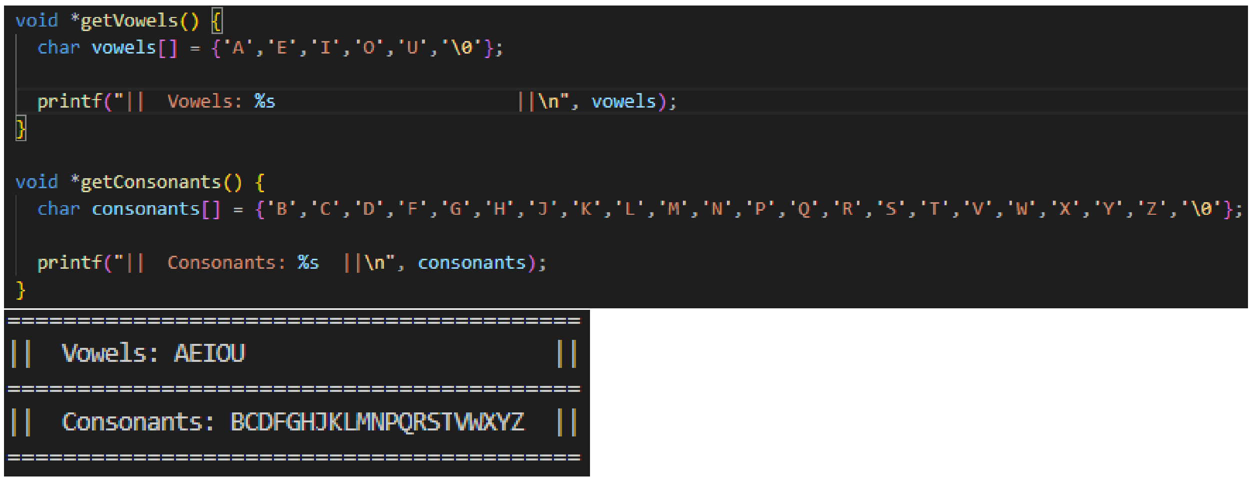


```c
 printf("==============================\n");
 printf("||<<<  Provide A String  >>>||\n");
 printf("==============================\n");
 printf("\nInput: ");
 scanf(" %99[^\n]", task2String );
 printf("===================\n");
 printf("|| [1] UpperCase ||\n");
 printf("|| [0] LowerCase ||\n");
 printf("===================\n");
 printf("\nSelect Case: ");
 scanf(" %d", &caseMode);
 printf("\n");
 changeCase(task2String, caseMode);
 printf("|| Converted String: %s ||\n",task2String);
 printf("\n");
void changeCase(char *sPtr, int mode) {
    while (*sPtr != '\0') {
        if (mode == 1) {
            *sPtr = toupper(*sPtr);
        } else if (mode == 0) {
            *sPtr = tolower(*sPtr);
        }
        sPtr++;
    }
}
```

```
==============================
||<<<  Provide A String  >>>||
==============================

Input: Rajel Johann Samano
===================
|| [1] UpperCase ||
|| [0] LowerCase ||
===================

Select Case: 0

|| Converted String: rajel johann samano ||
```

The task was to **"Display character strings according to the function changeCase(char *sPtr,int mode) according to the following (If mode == 1 , upper case; If mode == 0, lower case)."** My approach to this was first copying the code that was given to as in the pdf file as you can get some ideas from there and after that I added a selection whether to have it lower or upper case.

Figure3. LA4_codeTasks.c with output



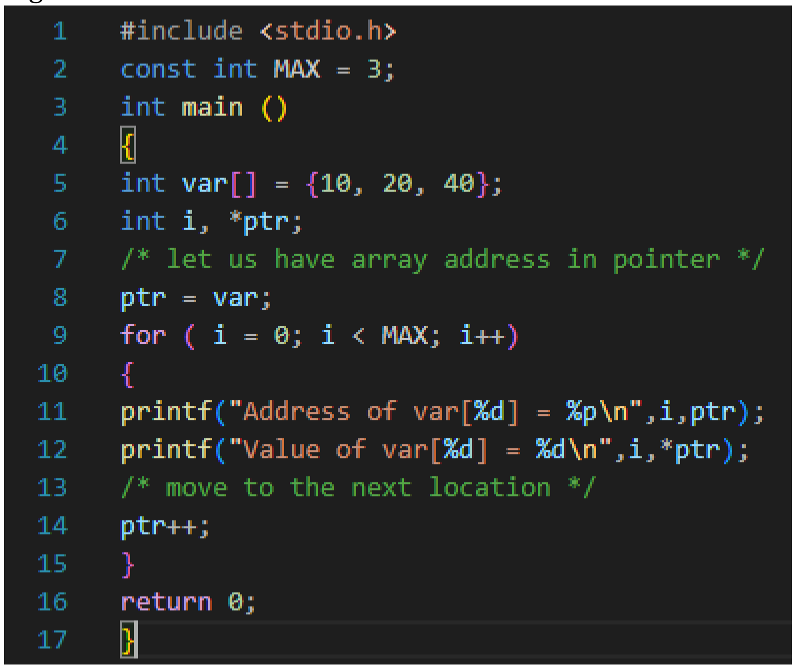
```c
void *getVowels() {
  char vowels[] = {'A','E','I','O','U','\0'};

  printf("||  Vowels: %s                    ||\n", vowels);
}

void *getConsonants() {
  char consonants[] = {'B','C','D','F','G','H','J','K','L','M','N','P','Q','R','S','T','V','W','X','Y','Z','\0'};

  printf("||  Consonants: %s  ||\n", consonants);
}
```

```
============================================
||  Vowels: AEIOU                         ||
============================================
||  Consonants: BCDFGHJKLMNPQRSTVWXYZ     ||
============================================
```

The task was to **"Display vowels A,E,I,O,U with *getVowels() and consonants with *getConsonants() in a program."** My approach to this was make a function with getVowels() and getConsonants() and declared variables vowels for vowels and consonants with consonants after I gave the value for each of them as follows for vowels = {'A','E','I','O','U','\0'} and consonants = {'B','C','D','F','G','H','J','K','L','M','N','P','Q','R','S','T','V','W','X','Y','Z','\0'} then after that I displayed it or used printf if there is no null it would work but displays it wrong.

III. EXPERIMENTAL FINDINGS / DISCUSSIONS

Figure 4. Exercise1



```
#include <stdio.h>
const int MAX = 3;
int main ()
{
int var[] = {10, 20, 40};
int i, *ptr;
/* let us have array address in pointer */
ptr = var;
for ( i = 0; i < MAX; i++)
{
printf("Address of var[%d] = %p\n",i,ptr);
printf("Value of var[%d] = %d\n",i,*ptr);
/* move to the next location */
ptr++;
}
return 0;
}
```

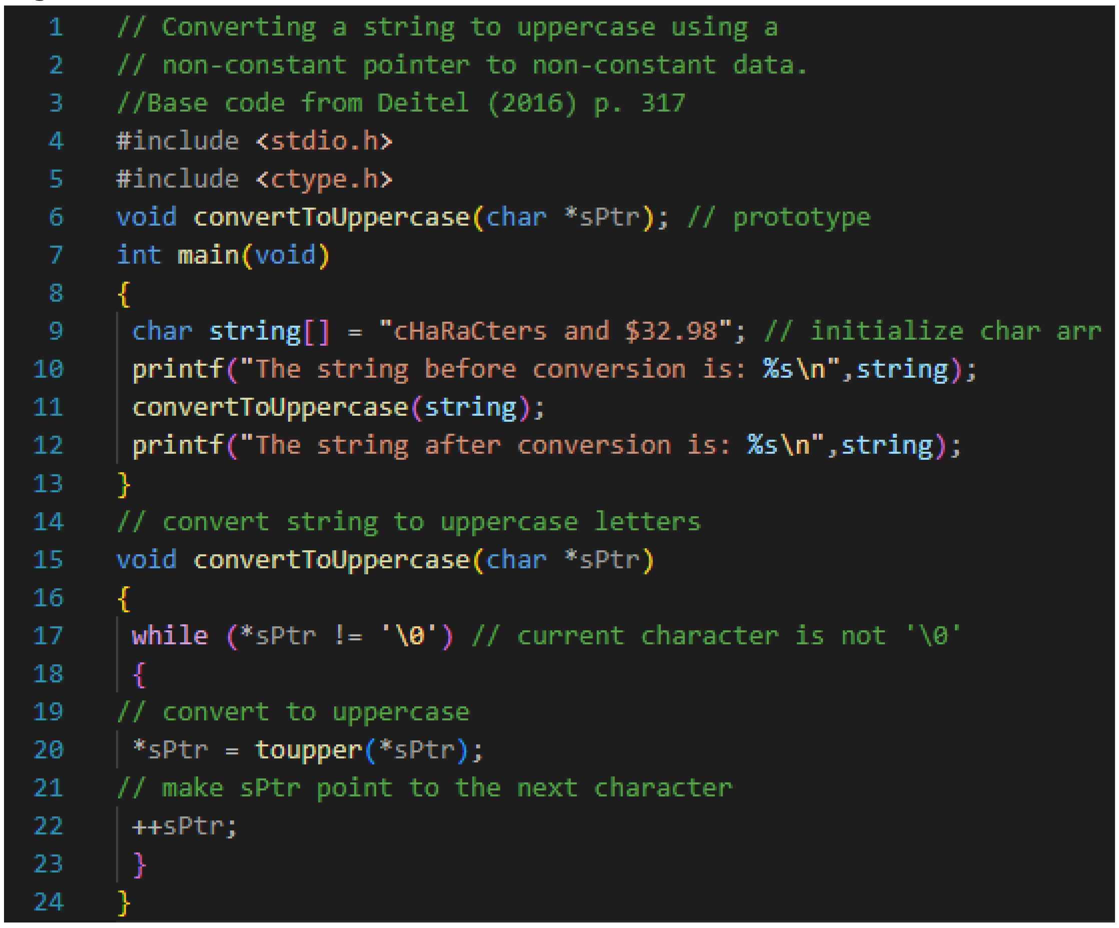
1. **What is the effect or result of applying arithmetic operations +, -, ++ and -- to pointers?**

Ans: The **(+)** moves the pointer elements in memory forward. The(-) moves the memory backward. The **(++)** Increments the pointer by one size of element forward while the decrements **(- -)** in backwards.

2. **Why is the '&' operator not required in array reference?**

Ans: For pointer '&' has no use as it behaves like a pointer already.

Figure 5. Exercise2



```c
// Converting a string to uppercase using a
// non-constant pointer to non-constant data.
//Base code from Deitel (2016) p. 317
#include <stdio.h>
#include <ctype.h>
void convertToUppercase(char *sPtr); // prototype
int main(void)
{
 char string[] = "cHaRaCters and $32.98"; // initialize char arr
 printf("The string before conversion is: %s\n",string);
 convertToUppercase(string);
 printf("The string after conversion is: %s\n",string);
}
// convert string to uppercase letters
void convertToUppercase(char *sPtr)
{
 while (*sPtr != '\0') // current character is not '\0'
 {
// convert to uppercase
 *sPtr = toupper(*sPtr);
// make sPtr point to the next character
 ++sPtr;
 }
}
```

1. What functions belong to ctype.h?

Ans: The functions that belong to ctype.h are:

•**isalnum()**This function identifies the alphanumeric characters.

•**isalpha()**This function identifies the alphabets from other characters.

•**isblank()**This function identifies the blank spaces from other characters.

•**iscntrl()**This function identifies the control characters(\n, \b, \t, \r).

•**isdigit()**This function identifies numbers in character.

•**islower()**This function identifies the lowercase alphabets.

•**isprint()**This function identifies the printable characters.

•**ispunct()**This function identifies punctuation characters (characters that are neither alphanumeric nor space).

•**isspace()**This function identifies white-space characters.

•**isupper()**This function identifies the uppercase alphabets.

•**isxdigit()**This function identifies the hexadecimal digit.

•**tolower()**This function converts uppercase alphabet to lowercase alphabet.

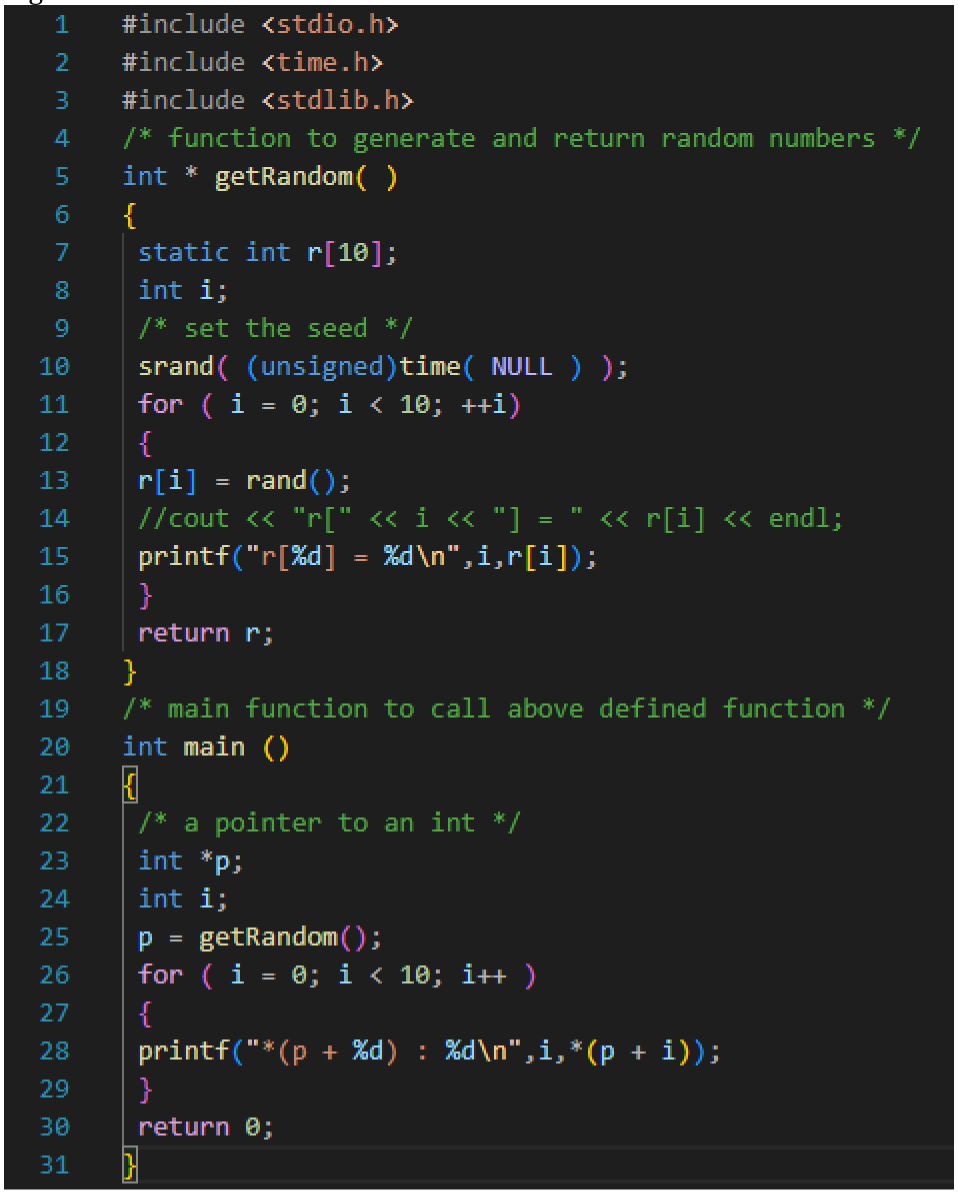
•**toupper()**This function converts lowercase alphabet to uppercase alphabet.

2. What is the purpose or meaning of '\0'?

Ans: The purpose or meaning of '\0' is that it's a null terminator it is used at the end of a string.

3. In what situations can pass by reference more favorable over pass by value?
Ans: Use pass by value when you are only "using" the parameter for some computation, not changing it for the client program In pass by reference (also called pass by address), a copy of the address of the actual parameter is stored. Use pass by reference when you are changing the parameter passed in by the client program.

Figure 6. Exercise3



```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
/* function to generate and return random numbers */
int * getRandom( )
{
 static int r[10];
 int i;
 /* set the seed */
 srand( (unsigned)time( NULL ) );
 for ( i = 0; i < 10; ++i)
 {
 r[i] = rand();
 //cout << "r[" << i << "] = " << r[i] << endl;
 printf("r[%d] = %d\n",i,r[i]);
 }
 return r;
}
/* main function to call above defined function */
int main ()
{
 /* a pointer to an int */
 int *p;
 int i;
 p = getRandom();
 for ( i = 0; i < 10; i++ )
 {
 printf("*(p + %d) : %d\n",i,*(p + i));
 }
 return 0;
}
```

1. What is the purpose of 'static' keyword?
Ans: The static keyword in c is used to specify that a variable or a function has static storage duration. A variable declared as static inside a function retains its value even after the function has returned, which means the variable remains in memory throughout the life of the program.

2. What can be observed from the results of running the code?
Ans: From had I observed it prints out random numbers and has a function *getRandom it uses the rand and srand for the generation of the random numbers.

IV. CONCLUSIONS

I learned a lot from this laboratory activity I didn't know there was a symbol that lets me have a variable that could be used anywhere and it can be changed through different ways or process to conclude this fourth lab activity I would say I know somewhat a good understanding of pointers.

References:

https://courses.washington.edu/css342/zander/css332/passby.html

https://datatrained.com/post/static-keyword-in-c/#:~:text=The%20static%20keyword%20in%20c%20is%20used%20to%20specify%20that,the%20life%20of%20the%20program.

https://www.geeksforgeeks.org/taking-string-input-space-c-3-different-methods/#:~:text=So%20we%20use%20%E2%80%9C%25%5B%5E,n%5Ds%E2%80%9D%2Cstr)%3B

https://www.programiz.com/c-programming/examples/vowel-consonant#google_vignette

https://stackoverflow.com/questions/50312194/how-to-print-a-char-array-in-c-through-printf

https://stackoverflow.com/questions/9053658/correct-format-specifier-to-print-pointer-or-address