United International University
Department of Computer Science and Engineering
CSI 124: Advanced Programming Laboratory
Lab Work # 10
Summer 2015

---

1 What is a pointer ?

We all know about variables and their data types. Every variable has a value and it is stored in a memory address by the compiler. Let **v** be a variable. We can get the address of **v** by using an '&' operator before **v**. Lets assign the address of **v** to another variable **pv** like **pv = &v**.
Here **pv** is a pointer variable that points to the address of **v**
How to declare a pointer ? Take a look:
data-type *ptvar
i.e. ***int *a , float *a , char *a***

```
#include<stdio.h>

int main()
{
    int v = 3;
    int q[]={3,4,5,6,3,2};
    int *pv;

    pv = &v;
    printf("*pv=%d v=%d\n", *pv, v);

    *pv = 0;
    printf("*pv=%d v=%d\n", *pv, v);

    pv = q;
    for(i = 0; i < 6; i++)
        printf("%d ",*(pv+i));

    return 0;
}
```

1. Now try to compile this code in the IDE and see the output.
2. WAP that will count the number of times a particular number is in an array using pointer.

2 Functions Again !

In this task, you will learn how to pass pointer to a function and get acquainted with the terms ***pass by value*** and ***pass by reference***. When we pass some value to a function using pointers and change the value in that function, the main value is affected.This is called ***pass by reference***. Whereas when we do the same

thing but not using pointers, the main value is not affected. This is called ***pass by value***. The difference between them and the mechanism of passing pointers are shown below:

```
#include<stdio.h>

void passByValue(int b,char s[]){
    b = 5;
    s[0] = 'F';
    printf("In passByValue and b = %d , s = %s\n",b,s);
}

void passByReference(int *b,char *t){
    *b = 5;
    t[0] = 'G';
    printf("In passByReference and b = %d, s = \n",*b);
    while(*t){
        printf("%c",*t);
        t++;
    }
    printf("\n");
}

int main(){

    int a = 4;
    int *p = &a;

    char str[40] = "No one beats the wiz";
    char *t = str;

    printf("Before call: %d %s\n", a, str);
    passByValue(a,str);
    printf("After first call: %d %s\n", a, str);
    passByReference(p,t); //passByReference(&a,t);
    printf("After second call: %d %s\n",a,str);

    return 0;
}
```

1. Now WAP that passes a pointer to an array to a function and prints the array in that function.
2. WAP that passes two array pointers to a function and prints '1' if the array elements are same and '0' if they are not. Function prototype would be:

   ```
   int sameArray(int *a, int *b, int size);
   ```

3 Memory Allocation

Till now whenever we needed an array, we had to declare the size of the array to a predetermined value. Most of the times the additional space is wasted as we do not use all the available indexes in an array. To overcome this, comes the idea of dynamic memory allocation. We can allocate memory space using the ***malloc*** function. Have a look at this code snap:

```
#include<stdio.h>

int main(){
    int i,n,*x;
    printf("how many numbers?\n");
    scanf("%d",&n);

    x = (int *)malloc(n*sizeof(int));
    for(i = 0;i < n; i++){
        scanf("%d",(x+i));
    }

    // here (x+i) is the address of ith index of the array

    printf("The array is:\n");
    for(i = 0;i < n; i++){
        printf("%d", *(x+i));
    }

    return 0;
}
```

1. WAP that copies an input string to a dynamically allocated memory space
2. Implement a **VECTOR** for storing int value. A **VECTOR** is an array that automatically doubles its size whenever it is full.

4 Practise Matches :)

1. Find the element with highest frequency in the array using pointer.
2. WAP that passes two strings as pointer to a function and prints '1' if they are equal and '0' id they are not.
3. Learn how to return a pointer from a function.
4. Learn allocating memory using **calloc** function.