

## Lab 9

### Theory:

The string.h header defines one variables type, one macro, and various functions for manipulating arrays of characters. The stdlib initializes the dynamic memory in c programming. It is written as stdlib.h. Some of the memory functions are described below

- **Malloc:** It allocates a block of memory in bytes at runtime.
- **Calloc:** It is used for allocating continuous block of memory at runtime.
- **Realloc:** Is is used for reducing or extending allocated memory space.
- **Free:** It deallocates previously used memory space.

### Methodology:

In the first program user was asked for the element and after that memory was allocated according to the memory space, malloc function was used in first program. Similarly in second program again user was asked for value and memory was allotted but in this program continuous block of memory was given to the program by system. In the third program first we provided the value then by using malloc we provided the memory and then we reallocated memory using calloc function then again deallocated the space using free function which helps in low memory consumption.

### Objectives:

1. To be familiar with the syntax and structure of C-programming.
2. To learn the problem solving technique.

3. To learn about dynamic memory allocation.

### Programs:

- Find out the errors and output of the given programs.

### Codes:

#### Program 1:

```
#include<stdio.h>

#include<stdlib.h>

int main(){

int* ptr;

int n,i;

printf("Enter numbers of elements:");

scanf("%d",&n);

ptr=(int*)malloc(n * sizeof(int));

if(ptr==NULL ){

    printf("memory not allotted");

    exit(0);

}

else{

    printf("memory sucessfully allotted using malloc");

    for(i=0;i<n;i++){

        ptr[i]=i+1;

    }

    printf("\nthe elements of array are:");

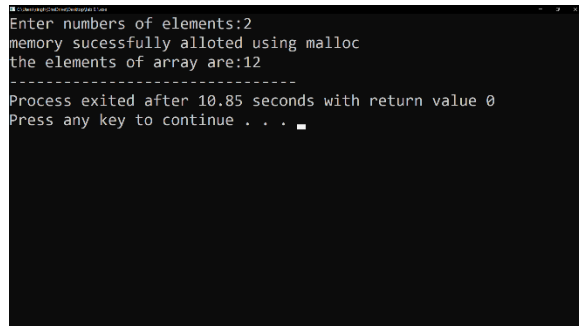
    for(i=0;i<n;i++){
```

```

        printf("%d",ptr[i]);
    }
}
return 0;
}

```

### **Output:**



```

Enter numbers of elements:2
memory successfully allotted using malloc
the elements of array are:12
-----
Process exited after 10.85 seconds with return value 0
Press any key to continue . . . 

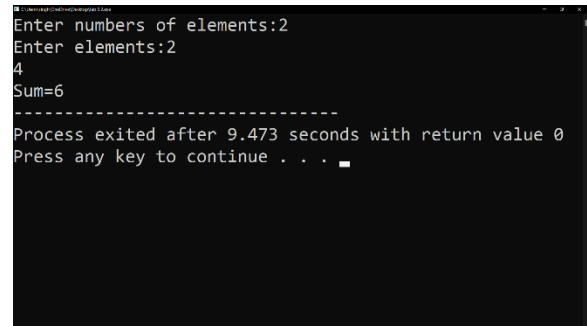
```

```

for(i=0;i<n;++i){
    scanf("%d",ptr+i);
    sum +=*(ptr+i);
}
printf("Sum=%d",sum);
free(ptr);
return 0;
}

```

### **Output:**



```

Enter numbers of elements:2
Enter elements:2
4
Sum=6
-----
Process exited after 9.473 seconds with return value 0
Press any key to continue . . . 

```

### **Program 2:**

```

#include<stdio.h>
#include<stdlib.h>
int main(){
    int *ptr,n,i,sum=0;
    printf("Enter numbers of elements:");
    scanf("%d",&n);
    ptr=(int*)calloc(n,sizeof(int));
    if(ptr==NULL ){
        printf("ERROR!      memory      not
alloted");
        exit(0);
    }
    printf("Enter elements:");

```

### **Program 3:**

```

#include<stdio.h>
#include<stdlib.h>
int main(){
    int* ptr;
    int n,i;
    n=5;
    printf("Enter number of elements:%d\n",n);
    ptr=(int*)calloc(n,sizeof(int));
    if(ptr=NULL){
        printf("memory not allocated.\n");
        exit(0);
    }

```

```

}
else{
    printf("memory sucesfully allocated
using calloc:\n");

    for(i=0;i<n;++i)
    {
        ptr[i]=i+1;
    }

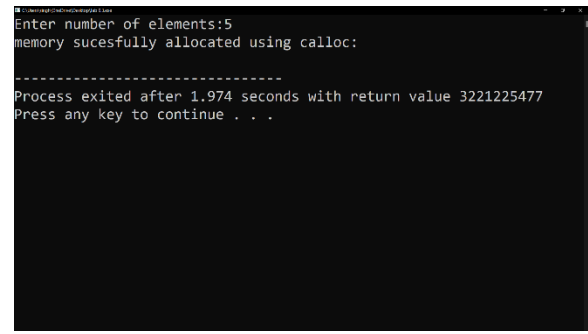
    printf("the elements of array are:");
    for(i=0;i<n;++i){
        printf("%d",ptr[i]);
    }
    n=10;
    printf("\n\nEnter the new size of the");
    ptr = realloc(ptr,n * sizeof(int));
    printf("memory sucessfully re-allocated");
    for(i=5;i<n;++i){
        ptr[i]=i+1;
    }

    printf("the elements of array are:");
    for(i=0;i<n;++i){
        printf("%d",ptr[i]);
    }
    free(ptr);
}

return 0;
}

```

## **Output:**



```

Enter number of elements:5
memory sucesfully allocated using calloc:

1 2 3 4 5
-----
Process exited after 1.974 seconds with return value 3221225477
Press any key to continue . . .

```

## **Discussion and conclusion:**

This is our 9<sup>th</sup> lab practical. The program is focused on finding the output. From this lab, I understood about dynamic memory allocation. Every code was correct and error free and no bugs were found and it showed a lot about c-programming and its structure.