B.S. Polytechnic Institute

Computer Technology

Subject: **Data Structure & Algorithm**

Subject Code: 6631

Job No:

Name of the job:

Name of the student:

Roll No:

Job Practicing Date:

Mark Distribution	
Practice	(6):
Interest on Practicing Works	(2):
Precaution & Cleanliness	(2):
Total	(10):

Signature of Teacher:

Experiment No: 01
Experiment Name: Search the Largest Element of list.

Theory: To store the list of elements in a array and find out the largest element of list. An array is the collection of homogeneous data items. The elements of an array are stored in successive memory locations. An array is referred by array name and index.

Algorithm:

1. Input :[1.. .. .. .n];
2. for(i=1;i<=n;i++);
   store data to x[i];
3. large = x[i];
4. for(i=x;i<=n;i++);
   if(x[i]>large), large = x[x];
5. Output: x is largest value (print the value large).

Program code:

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int mat[20];
    int s,max;
    printf("\nSize of your Array is : ");
    scanf("%d",&s);
    printf("\nEnter your %d element of an array : \n", s);
    for(int i=0;i<s;i++)
    {
        scanf("%d",&mat[i]);
    }
    max=mat[0];

    for(int i=1;i<s;i++)
    {
        if(max < mat[i])
        max = mat[i];
    }
    printf("\nmaximum value element from given %d element is : %d",s,max);
    getch();
    return 0;}
```

Conclusion: x is the array to store data and n is size of the list. Above the code is executed successfully by turbo c compiler.

Experiment No: 02
Experiment Name: To insert the element into the array at a given position.

Theory: Here it is assumed that all the cells of an array A contain data. So the data from the passions m to n of A are copied to the positions m+1 to n+1 of the array A is empty and the element to be inserted in placed at the position.

Algorithm:

1 Input A :[1.. .. .. .n]; The position of insertion m and the data x;
2 .Increase the size of the array, A [1… … … n+1];
3 for(i=m;i<=n;i++);
A [i+1] = A[i];
4 A[m] = x;
5 Output: The array A with size n +1;

Programming Code:

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

int main(){
   int arr[100] = {10, 5, 46, 2, 100, 97};
   int n, max, i;
   n = 0;

   clrscr();
   if(i > n){
      printf("postion you wanted to insert");
      scanf("%d", &n);
   }

   printf("Value for position: \n");
   scanf("%d", &max);

   for(i=7; i>n-1; i--){
      arr[i+1] = arr[i];
      arr[n] = max;
   }

   printf("Array after insertion: \n");
   for(i = 0; i < 7; i++){
      printf("%d\t", arr[i]);
   }

   getch();
}
```

Conclusion: This program is executed successfully by turbo c compiler.

Experiment No: 03

Experiment Name: Given a list of data elements arranged in ascending or descending order locate a particular element from the list(Binary Search)

Theory: Suppose that we are given a list of elements arranged in ascending or descending order and a target element. In binary searching, at first we have to identify the middle element of the list and compare this middle element with target element. If middle element is the target element then searching is successful and terminated.

Algorithm:

```
BinarySearch(A[0..N-1], value, low, high) {
   if (high < low)
      return -1 // not found
   mid = low + ((high - low) / 2)
   if (A[mid] > value)
      return BinarySearch(A, value, low, mid-1)
   else if (A[mid] < value)
      return BinarySearch(A, value, mid+1, high)
   else
     return mid // found
   }
```

Program code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[20],n,i,top,bottom,mid,f,s;
clrscr();
printf("Enter the value of n:");
scanf("",&n);
printf("Enter the values:\n");
for(i=0;i<n;i++)
{
scanf("",&a[i]);
}
printf("Enter the element to be searched:");
scanf("",&s);
top=0;
bottom=n-1;
f=0;
while(top<=bottom && f==0)
{
mid=(top+bottom)/2;
if(s>a[mid])
{top=mid+1;
}
else if(s<a[mid])
{
bottom=mid-1;
}else
```

```
f=1;}
if(f==1)
printf("Element found");
else
printf("Element not found.");
getch();
}
```

Conclusion: This program is executed successfully by turbo c compiler.

Experiment No: 04
Experiment Name: Pus operation in a stack.

Theory: Here, top is an indicator indicates the top of the stack and item is an element to be added to the stack. M is the of the stack. Overflow occurs when we try to insert an element into the stack which is already full.

On the other, here top is an indicator indicates the top element of the stack, M is the size of the array and x is a variable where we access top element of the stack.

Pus Algorithm:

1. Declare the stack and top;
   Stack[1… … … M], top:
2. Add an item in the stack:

   If(top<M)

   {
   Top = top + 1;
   Stack [top] = item;
   }
   Else print" Overflow";
3. Output: update stack.

Programming code:

```
#define MAX 4 //you can take any number to limit your stack size
#include<stdio.h>
#include<conio.h>

int stack[MAX];
int top;

void push(int token)
{
   char a;
   if(top==MAX-1)
   {
      printf("Stack full");
      return;
   }
   do
   {
      printf("\nEnter the token to be inserted:");
      scanf("%d",&token);
      top=top+1;
      stack[top]=token;
      printf("do you want to continue insertion Y/N");
      a=getch();
   }
   while(a=='y');
}
```

```
int pop()
{
    int t;
    if(top==-1)
    {
        printf("Stack empty");
        return -1;
    }
    t=stack[top];
    top=top-1;
    return t;
}

void show()
{
    int i;
    printf("\nThe Stack elements are:");
    for(i=0;i<=top;i++)
    {
        printf("%d",stack[i]);
    }
}

int main()
{
    char ch , a='y';
    int choice, token;
    top=-1;
    printf("1.Insert");
    printf("\n2.Delete");
    printf("\n3.show or display");
    do
    {
        printf("\nEnter your choice for the operation: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
            push(token);
            show();
            break;
            }
            case 2:
            {
            token=pop();
            printf("\nThe token deleted is %d",token);
            show();
            break;
            }
            case 3:
            {
```

```
            show();
            break;
            }
            default:printf("Wrong choice");
            break;
        }
        printf("\nDo you want to continue(y/n):");
        ch=getch();
    }
    while(ch=='y'||ch=='Y');
    getch();
}
```

Conclusion: This program is executed successfully by turbo c compiler.

Experiment No: 05

Experiment Name: Pop operation in a stack.

Theory: Here, top is an indicator indicates the top of the stack and item is an element to be added to the stack. M is the of the stack. Overflow occurs when we try to insert an element into the stack which is already full.

On the other, here top is an indicator indicates the top element of the stack, M is the size of the array and x is a variable where we access top element of the stack.

Pop Algorithm:

1. Declare the stack and top;
   Stack [1… … … M], top:
2. Access the top element:

```
If(top == 0) print “stack is empty”;
else
{
        X = Stack [top];
        Top = top -1;
        }
```

3. Output: update stack list.

Programming code:

```
#define MAX 4 //you can take any number to limit your stack size
#include<stdio.h>
#include<conio.h>

int stack[MAX];
int top;

void push(int token)
{
   char a;
   if(top==MAX-1)
   {
      printf("Stack full");
      return;
   }
   do
   {
      printf("\nEnter the token to be inserted:");
      scanf("%d",&token);
      top=top+1;
      stack[top]=token;
      printf("do you want to continue insertion Y/N");
      a=getch();
   }
   while(a=='y');
}
```

```
int pop()
{
    int t;
    if(top==-1)
    {
        printf("Stack empty");
        return -1;
    }
    t=stack[top];
    top=top-1;
    return t;
}

void show()
{
    int i;
    printf("\nThe Stack elements are:");
    for(i=0;i<=top;i++)
    {
        printf("%d",stack[i]);
    }
}

int main()
{
    char ch , a='y';
    int choice, token;
    top=-1;
    printf("1.Insert");
    printf("\n2.Delete");
    printf("\n3.show or display");
    do
    {
        printf("\nEnter your choice for the operation: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
            push(token);
            show();
            break;
            }
            case 2:
            {
            token=pop();
            printf("\nThe token deleted is %d",token);
            show();
            break;
            }
            case 3:
```

```
            {
            show();
            break;
            }
            default:printf("Wrong choice");
            break;
        }
        printf("\nDo you want to continue(y/n):");
        ch=getch();
    }
    while(ch=='y'||ch=='Y');
    getch();
}
```

Conclusion: This program is executed successfully by turbo c compiler.