**Shri Vaishnav Vidyapeeth Vishwavidyala, Indore**

**Shri Vaishnav Institute of Information Technology**

**Department of Computer Science & Engineering**



**LAB FILE**

**Jan-Jun 2023**

**Name of the Student: JHILMIL GEETE**

**Enrollment: 2210DMTCSE12010**

**Program: B.Tech+M.Tech(CSE)**

**Section: H**

**Year/Sem: I / II**

**Subject Code:** **BTCS201N**

**Subject Name:** **Data Structure and Algorithms**

**Name of Subject** **Teacher: Dr. Sandeep Kumar Jain**

**INDEX**

NAME-JHILMIL GEETE

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Aim/Objective** | **Date of Experiment** | **Sign /**  **Remarks** |
| 1 | WAP to find maximum and minimum number in an array. | 14/03/2023 |  |
| 2 | Perform searching and sorting in an array. | 14/04/2023 |  |
| 3 | WAP to find lowercase, uppercase, wordcount, total characters and numericdigits in a string. | 21/04/2023 |  |
| 4 | Perform insertion and deletion operation in 1D array. | 21/04/2023 |  |
| 5 | WAP to perform various operation in the single linked list.   1. CreateFirstNode 2. InsertAtBeginning 3. InsertAtLast 4. DeleteFromBeginning 5. DeleteFromLast 6. DisplayList | 26/04/2023 |  |
| 6 | WAP to perform insertion and deletion operation in circular linked list. | 04/05/2023 |  |
| 7 | Write a menu driven program to implement various operations as push, pop, display, isFull and isEmpty in a stack with the help of static memory allocation. | 11/05/2023 |  |
| 8 | Write a menu driven program to implement various operations as push, pop, display, isFull and isEmpty in a stack with the help of dynamic memory allocation. | 18/05/2023 |  |
| 9 | WAP for Tower of Hanoi using recursion. | 25/05/2023 |  |
| 10 | Illustrate queue implementation using array with following operation as enQueue, deQueue, isEmpty, displayQueue. | 09/06/2023 |  |
| 11 | Illustrate queue implementation using linked list with following operation as enQueue, deQueue, isEmpty, displayQueue. | 09/06/2023 |  |
| 12 | WAP to construct a binary search tree and perform deletion and inorder traversal on it. | 16/06/2023 |  |
| 13 | WAP to develop an algorithm for binary search, implement and test it. | 16/06/2023 |  |
| 14 | WAP for implementation of Bubble Sort. | 23/06/2023 |  |

ENROLLMENT NO. – 2210DMTCSE12010

|  |  |  |  |
| --- | --- | --- | --- |
| 15 | WAP for implementation of Insertion Sort. | 20/06/2023 |  |

# EXPERIMENT NO.-01

AIM:- WAP to find maximum and minimum number in an array.

DATE:- 29/03/2023

TOOL:- Dev C++

## ALGORITHM-

1. Initialize two variables, maxNum and minNum, with the first element of the array.
2. Iterate through the remaining elements of the array.
3. For each element:

If the element is greater than the current max, update max with the new value.

If the element is smaller than the current min, update min with the new value.

1. After iterating through all the elements, max will contain the maximum number, and min will contain the minimum number.

SOURCE CODE:-

//WAP to find maximum and minimum number in an array.

#include <stdio.h> int main() { int arr[5] = {4,8,12,14,2}; int max = arr[0]; int min = arr[0];

int i; for(i = 1; i < 5; i++){ if (arr[i] > max){ max = arr[i];

}

if (arr[i] < min) { min = arr[i];

}

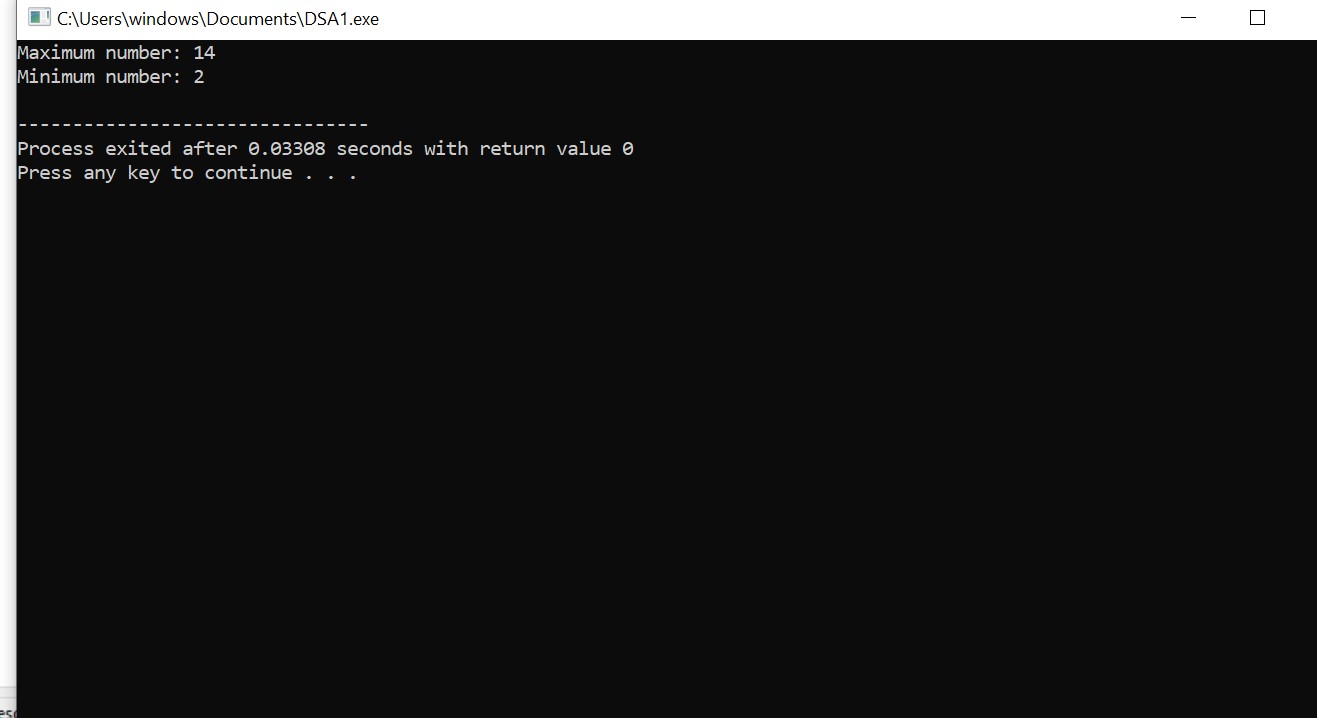
}

printf("Maximum number: %d\n", max); printf("Minimum number: %d\n", min);

return 0;

}

## OUTPUT-



## 

## EXPERIMENT NO. 02

AIM :- Write a program for performing searching and sorting in array.

DATE :- 05/04/2023

TOOL :- Dev C++

### ALGORITHM of Searching :- Step 01: Start

Step 02: [Initialize counter variable. ] Set i = 0

Step 03: Repeat Step 04 and 05 for i = 0 to i < n

Step 04: if a[i] = x, then jump to step 07

Step 05: [Increase counter. ] Set i = i + 1

Step 06: [End of step 03 loop. ]

Step 07: Print x found at i + 1 position and go to step 09

Step 08: Print x not found (if a[i] != x, after all the iteration of the above for loop. )

Step 09: Stop

SOURCE CODE :- 1. Searching

#include<stdio.h> #define MAX\_SIZE 50 int main()

{

int arr[MAX\_SIZE]; //defining the array int size,i,tosearch,found; printf("enter the size of array => "); scanf("%d", &size); printf("ente elements in array : "); for(i = 0; i< size; i ++)

{

scanf("%d", &arr[i]);

}

printf("/n enter element to search : "); scanf("%d", &tosearch); found = 0; for(i = 0; i <size; i++)

{

if(arr[i] == tosearch)

{

found = 1; break;

}

}

if(found == 1)

{

printf("entered element %d is found at position %d", tosearch, i+1);

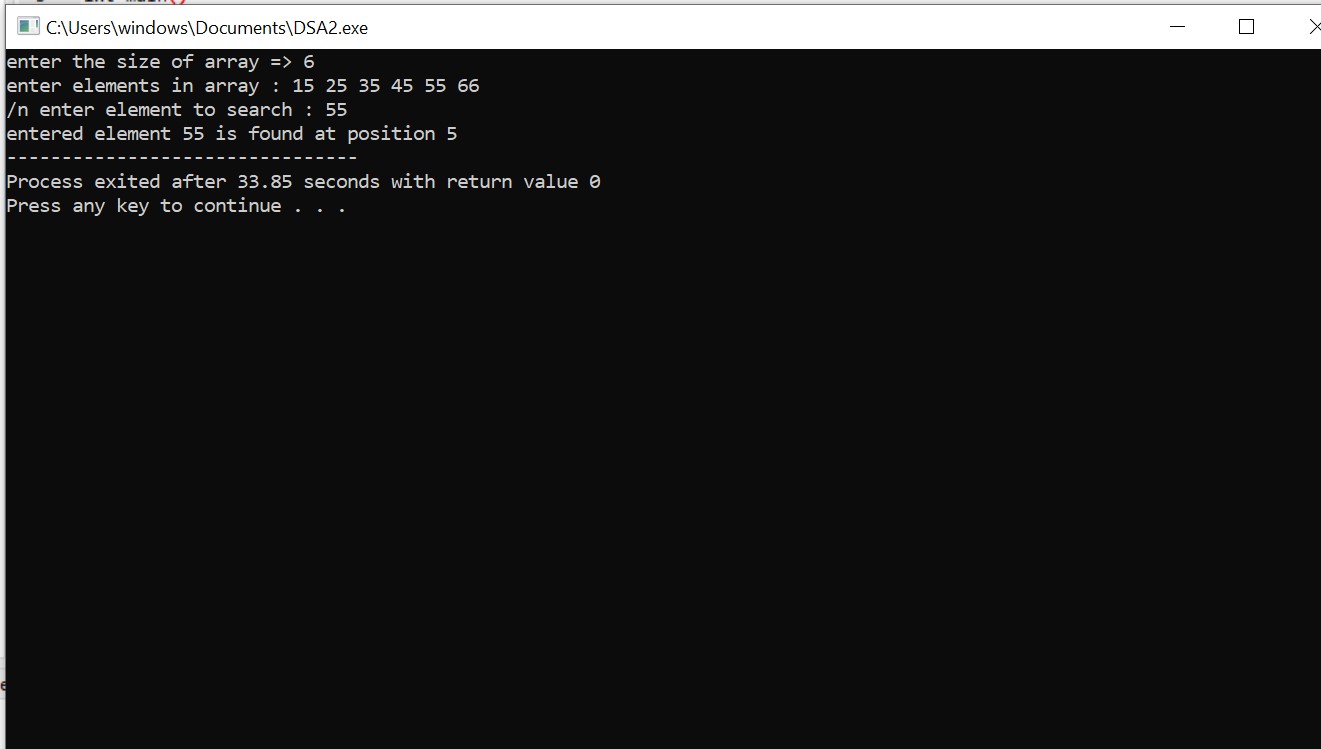
}

else{ printf("\n entered element %d is not found in array", tosearch);

} return 0;

}

### OUTPUT:-



#### 

#### ALGORITHM of Sorting:-

1. Create an array of fixed size (maximum capacity), lets say 10.
2. Take n, a variable which stores the number of elements of the array, less than maximum capacity of array.
3. Iterate via for loop to take array elements as input, and print them.
4. The array elements are in unsorted fashion, to sort them, make a nested loop.
5. In the nested loop, the each element will be compared to all the elements below it.
6. In case the element is smaller than the element present below it, then they are interchanged
7. After executing the nested loop, we will obtain an array in descending order arranged elements.

#### SOURCE CODE :- 2. Sorting

#include<stdio.h> void main()

{

int arr[20]; // define an array

int i,j,a,n;

printf("enter number of element in an array => "); scanf("%d", &n); printf("enter the elements of array :"); for(i = 0; i<n; i++) scanf("%d", &arr[i]); for(i = 0; i<n; i++)

{

for(j = i+1; j<n; j++)

{ if(arr[i] > arr[j])

{ a = arr[i]; arr[i] = arr[j]; arr[j] = a;

}

}

}

printf("the number in ascending order is : "); for(i = 0; i < n; i++)

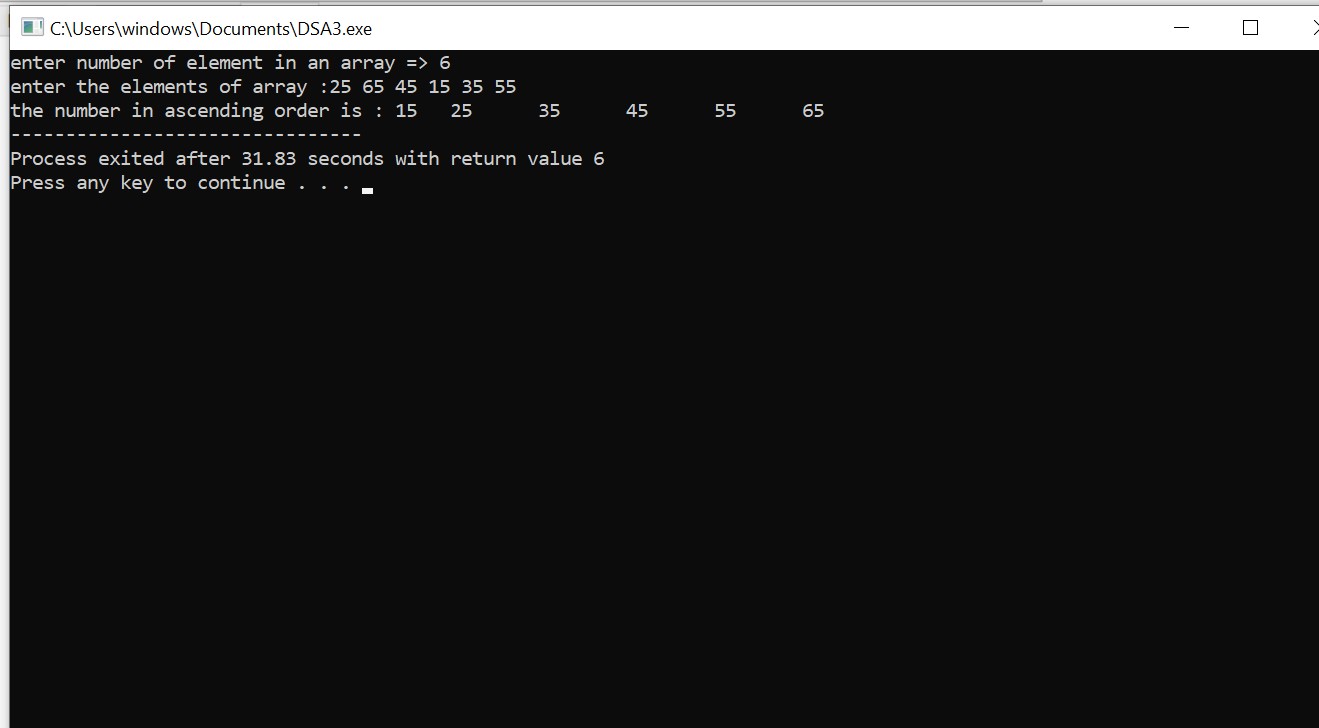
{

printf("%d\t",arr[i]);

}

}

### OUTPUT:-



## 

## EXPERIMENT NO.:-03

AIM:- WAP to find lowercase, uppercase, wordcount, total characters and numericdigits in a string.

DATE:- 12/04/2023

TOOL:- Dev C++

### ALGORITHM:-

1. Scan string str from 0 to length-1.
2. check one character at a time

* if(str[i] >= ‘A’ and str[i] <=’Z’), then it is uppercase letter,
* if(str[i] >= ‘a’ and str[i] <=’z’), then it is lowercase letter, • if(str[i] >= ‘0’ and str[i] <=’9’), then it is number, else it is a special character

3. Print all the counters

### SOURCE CODE :-

//WAP to find lowercase, uppercase, wordcount, total characters and numericdigits in a string.

#include <stdio.h> #include <stdlib.h> int main()

{

char str[50];

int i;

int upper=0,lower=0,num=0,special=0,

chars=0;

printf("Please enter the string \n");

gets(str); i=0; for( i = 0; str[i] != '\0'; i++) { //check for uppercase if(str[i]>='A' && str[i]<='Z') { upper++;

//ceck lower case }else if(str[i]>='a' && str[i]<='z') { lower++;

//Check numeric }else if(str[i]>='0' && str[i]<='9') { num++;

}

else{//check special character special++;

}

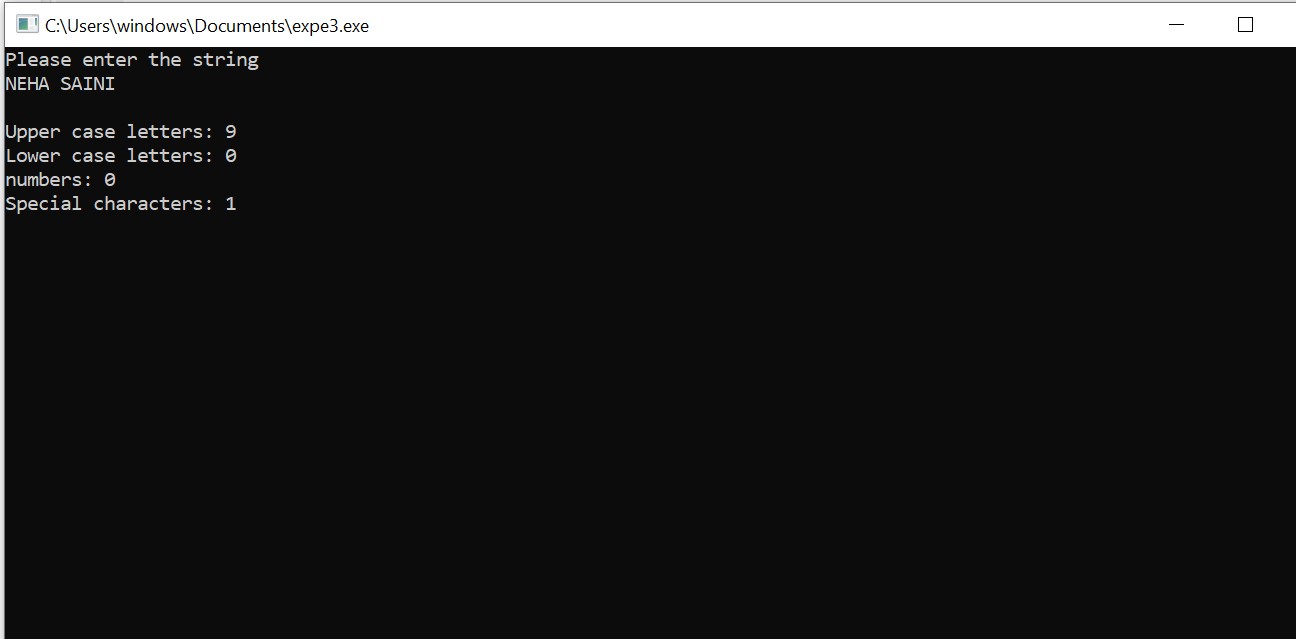
chars++;

}

printf("\nUpper case letters: %d",upper); printf("\nLower case letters: %d",lower); printf("\nnumbers: %d",num); printf("\nSpecial characters: %d",special); printf("\nTotal characters: %d",chars); return 0;

}

### OUTPUT:-



EXPERIMENT NO.-04

AIM:- Perform insertion and deletion operation in 1D array.

DATE:- 19/04/2023

TOOL:- Dev C++

### ALGORITHM OF DELETION :-

Step 01: Start

Step 02: [Initialize counter variable. ] Set i = index - 1

Step 03: Repeat Step 04 and 05 for i = index - 1 to i < size

Step 04: [Move ith element backward (left). ] set arr[i] = arr[i+1]

Step 05: [Increase counter. ] Set i ++;

Step 06: [End of step 03 loop. ]

Step 07: [Increase size of the array. ] set size++;

Step 08: Stop

### SOURCE CODE :-

//Perform deletion operation in 1D array.

#include <stdio.h> int main() { int i,arr[50], size, index; printf("Enter the size of the array: "); scanf("%d", &size); printf("Enter %d elements:\n", size

for (i = 0; i < size; i++) { scanf("%d", &arr[i]);

}

printf("Enter the index of the element you want to delete: "); scanf("%d", &index);

// Shift all elements after the index to the left for (i = index; i < size - 1; i++) { arr[i] = arr[i + 1];

}

// Decrease the size of the array by 1

size--;

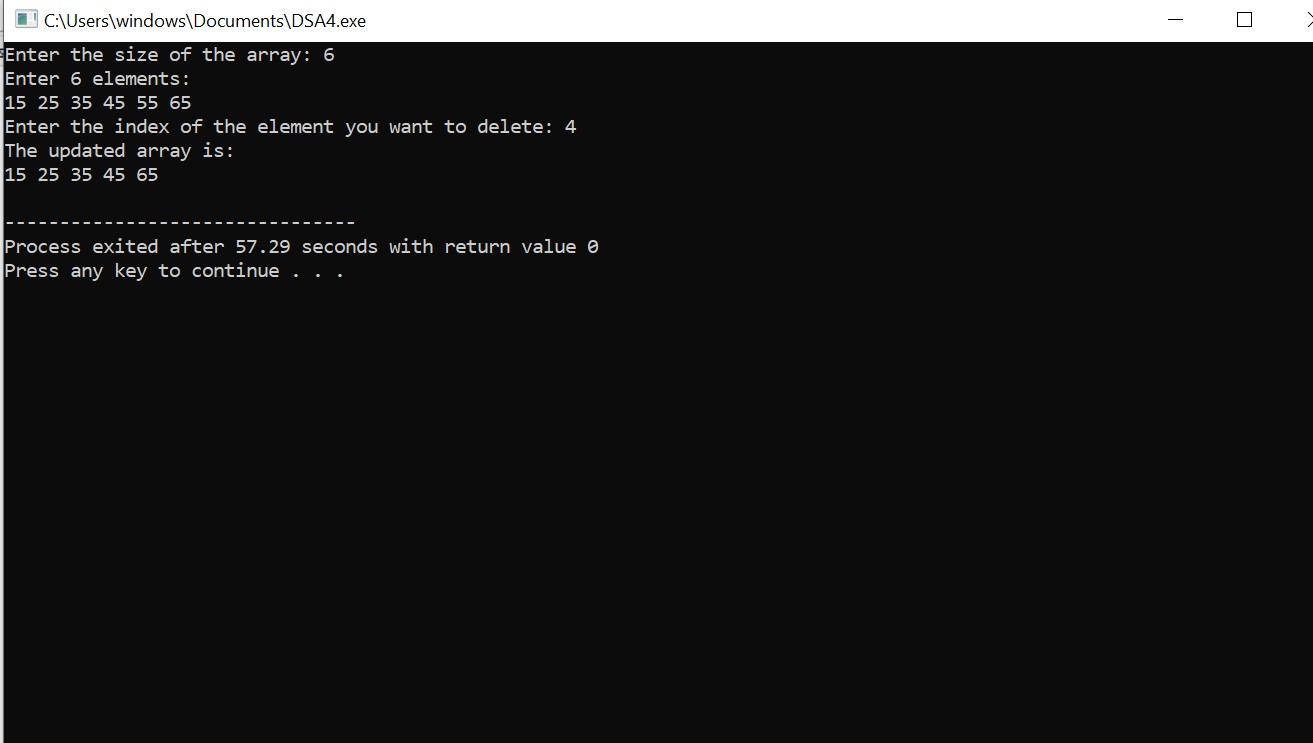
printf("The updated array is:\n"); for (i = 0; i < size; i++) { printf("%d ", arr[i]);

}

printf("\n"); return 0;

}

OUTPUT:-



### ALGORITHM OF INSERTION:-

AT BEGINNING –

1.First get the element to be inserted.

2.Increase size of array.

3.Then right sift all element of array a[i-1]=a[i-2].

4.Then insert the new element at index 0.

AT THE END –

1.First get the element to be inserted.

2.Increase size of array.

3.Then insert the new element at index of size-1.

At any index –

1. First get the element to be inserted, say x.
2. Then get the position at which this element is to be inserted, say index and increase the size of array.
3. Then shift the array elements from this position to one position forward(towards right), and do this for all the other elements next to index.
4. Insert the element x now at the position index, as this is now empty.

#### SOURCE CODE:-

#include <stdio.h> int main()

{ int size, i, new; printf("enter the size of array\n"); scanf("%d", &size); int a[size];

printf("enter the elements in array \n"); for (i = 0; i < size; i++)

{

scanf("%d", &a[i]);

}

//At the beginning printf("Insert the new element at beggining\n"); scanf("%d",&new);

size++; for(i=size; i>1; i--)

{

a[i-1]=a[i-2];

}

a[0]=new; //At the end printf("Insert the new element at end\n"); scanf("%d",&new);

size++; a[size-1]=new; //At any position int index,element; printf("Enter the position where you want to insert the new element\n"); scanf("%d",&index); printf("Enter the new element\n"); scanf("%d",&element);

size ++;

for(i=size-1; i>=index; i--){ a[i+1]=a[i];

}

a[index]=element; printf("Resultant array elements\n");

for (i = 0; i < size; i++)

{

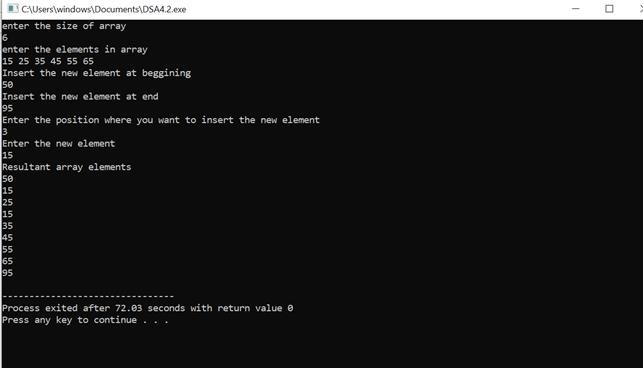
printf("%d\n", a[i]);

}

return 0;

}

### OUTPUT:-



## EXPERIMENT NO.-05

AIM:- WAP to perform various operation in the single linked list.

1. CreateFirstNode
2. InsertAtBeginning
3. InsertAtLast
4. DeleteFromBeginning
5. DeleteFromLast
6. DisplayList

DATE:- 26/04/2023

TOOL:- Dev C++

ALGORITHM:- 1. Declare a structure which contain data , and a pointer(next).

1. Declare a starting point as head.
2. For create first node
   1. Take a pointer and assign DMA.
   2. new-> data = data; iii. new->next=head; iv. assign new to head
3. For insertion at start
   1. take a new pointer and assign DMA.
   2. new->data=data; iii. new->next=head; iv. assign new to head;
4. For insert at end
5. take a new pointer and assign DMA.
6. take another pointer which start from head.
7. new->data= data; iv. traverse the list untill you reach the last node.
8. set the next pointer of the last node to new node.
9. new-> next = NULL;

6. Delete at start

1. take a pointer (ptr) which start from head.
2. head=head->next; iii. And then free(ptr);

7. Delete at end.

1. take pointer (p) which start from head.
2. take another pointer (q) which start from head->next. iii. traverse the list untill a reach last node.
3. assign p-> next=NULL;
4. free (q); 8. For print
5. take a pointer (ptr).
6. traverse list from head to last node. iii. and every time print element.

iv. ptr=ptr->next

SOURCE CODE:-

#include<stdio.h>

#include<stdlib.h>

struct node\* deleteFromBeginning(struct node\* head);

struct node\* deleteFromLast(struct node\* head);

//to create node

struct node

{

    int data;

    struct node \*next;

    int \*p;

};

//To display linklist

void tra(struct node\*ptr)

{

    while(ptr!=0) {

    printf("\nvalue is %d",ptr->data);

    ptr=ptr->next;

    }

}//insert at first

struct node\*insertatfirst(struct node \*head,int data)

{

struct node\*ptr=(struct node\*)malloc(sizeof(struct node));

    ptr->next=head;

    ptr->data=data;

    return ptr;

}

//insert at between

struct node\*insertatbtw(struct node\*head,int data,int index)

{

struct node\*ptr=(struct node\*)malloc(sizeof(struct node));

struct node\*p=head;

int i=0;

while(i!=index-1)

{

    p=p->next;

    i++;

}

    ptr->data=data;

    ptr->next=p->next;

    p->next=ptr;

    return head;

}

//insert at end

struct node\*insertatend(struct node\*head,int data)

{

struct node\*ptr=(struct node\*)malloc(sizeof(struct node));

ptr->data=data;

struct node\*p=head;

while(p->next!=NULL){

    p=p->next;

    }

p->next=ptr;

ptr->next=0;

return head;

}

// Delete the first node

struct node\* deleteFromBeginning(struct node\* head)

{

    if (head == NULL)

        return NULL;

    struct node\* temp = head;

    head = head->next;

    free(temp);

    return head;

}

// Delete the last node

struct node\* deleteFromLast(struct node\* head)

{

    if (head == NULL)

        return NULL;

    if (head->next == NULL) {

        free(head);

        return NULL;

    }

    struct node\* temp = head;

    while (temp->next->next != NULL)

        temp = temp->next;

    free(temp->next);

    temp->next = NULL;

    return head;

}

int main()

{

struct node \*head;

struct node \*second;

struct node \*third;

struct node\*fourth;

head=(struct node\*)malloc(sizeof(struct node));

second=(struct node\*)malloc(sizeof(struct node));

third=(struct node\*)malloc(sizeof(struct node));

fourth=(struct node\*)malloc(sizeof(struct node));

head->data=4;

head->next=second;

second->data=8;

second->next=third;

third->data=10;

third->next=fourth;

fourth->data=16;

fourth->next=NULL;

printf("created nodes:");

tra(head);

printf("\ninsertion at first");

head=insertatfirst(head,40);

tra(head);

printf("\ninsertion at between");

head=insertatbtw(head,26,2);

tra(head);

printf("\ninsertion at end");

head=insertatend(head,6);

tra(head);

printf("\ndeletion from beginning");

head = deleteFromBeginning(head);

tra(head);

printf("\ndeletion from last");

head = deleteFromLast(head);

tra(head);

return 0;

}

### OUTPUT:-

### 

## EXPERIMENT NO.-06

AIM:- WAP to perform insertion and deletion operation in circular linked list.

DATE:- 10/05/2023

TOO:-Dev C++

ALGORITHM:- 1)Declare a structure which contain data field and a pointer(next).

2)Create a pointer (tail)=NULL.

3)Create first node

1.take a pointer and assign memeory using malloc function(dynamic memory allocation).

2.Set a pointer new -> data = data;

3.The next field of new point its self.

4)Insert at start

1..take a pointer and assign memeory using malloc function(dynamic memory allocation).

2.new->data=data;

3.new->next=tail; 4.tail->next=new;

5)Update the tail of the list to point to the new.

6)Insert at end

1.allocate memory for the new node.

2.assign data to new node.

3.set the next pointer of the new to the next of the tail of the list.

4.set the next of the tail to new.

5.assign new to tail.

6)Delete at first

1.create a pointer (p) and intialise it to the next of the tail.

2.assign p->next to the tail->next.

3.then free to p.

7)delete at end 1. take two pointer p=tail->next; q=tail->next->next;

1. traverse the list unitl q do not reach tail.
2. assign p->next to tail->next.
3. assign next of q is NULL.
4. free(q);
5. assign p to tail.

8. for Traverse the list

1. first check it tail->next=NULL,so list is empty.
2. take a pointer(temp) which is start from next of tail.
3. traverse the list unitl temp not equal tail->next.
4. print the element of list.

### SOURCE CODE:-

//All operation of circular linked list

#include<stdio.h> #include<stdlib.h> struct node { int data; struct node \* next;

};

typedef struct node node; node\* createfirstnode(int data)

{

node \*new; new=(struct node\*)malloc(sizeof(struct node)); new->data=data; new->next=new; return new;

}

node\*insert\_at\_start(node\*tail,int data)

{

node\*new =(node\*)malloc(sizeof(node)); new->data=data; new->next=tail; tail->next=new; return tail;

}

node\*insert\_at\_end(node\*tail,int data)

{

node\*new =(node\*)malloc(sizeof(node)); new->data=data; new->next=tail->next; tail->next=new; tail=new; return tail;

}

node\*delete\_at\_first(node\*tail) { node\*p=tail->next; tail->next=p->next; free(p); return tail;

}

node\*delete\_at\_last(node\*tail) { node\*p=tail->next; node\*q=tail->next->next; while(q!=tail) { p=p->next; q=q->next;

}

p->next=tail->next; q->next=NULL;

free(q); tail=p; return tail;

}

node\*printlist(node\*tail) { if(tail->next==NULL){ printf("There is no node in linked list\n"); return 0;

}

node\*temp=tail->next;

do {

printf("%d\n",temp->data); temp=temp->next; } while(temp!=tail->next); return temp;

}

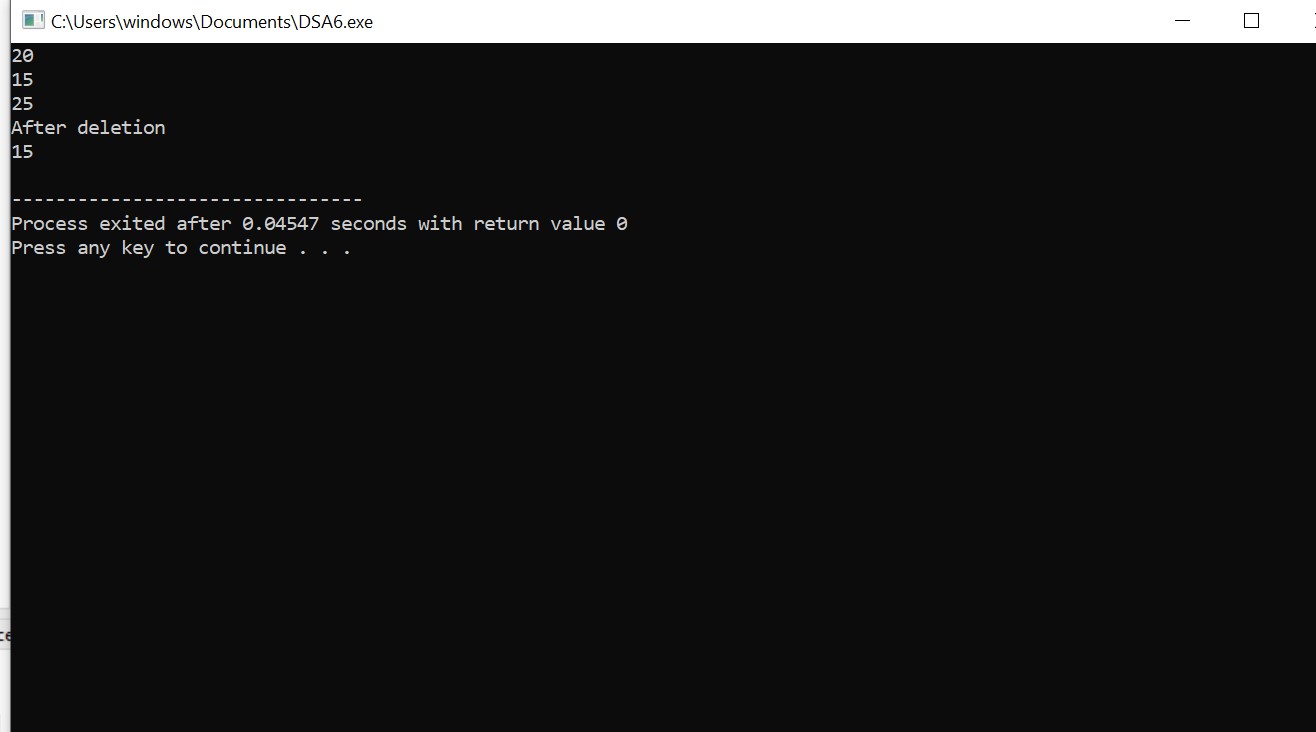
int main()

{

struct node \*tail=NULL; tail=createfirstnode(15); tail=insert\_at\_start(tail,20); tail=insert\_at\_end(tail,25); printlist(tail); printf("After deletion\n"); tail=delete\_at\_first(tail); tail=delete\_at\_last(tail); printlist(tail); return 0;

}

#### OUTPUT:-



## EXPERIMENT NO.-07

AIM:- Write a menu driven program to implement various operations as push, pop, display, isFull and isEmpty in a stack with the help of static memory allocation.

DATE:- 17/05/2023

TOOL:- Dev C++

ALGORITHM:- 1. Define size 5;

1. Initialize global vairiable top = -1 and integer arr.
2. Push

1 First we check the stack is full or not .

2. If empty so we increment top to top +1 and insert data at top.

4. Pop

1. First we check the stack in empty or not.
2. If it is not empty we do pop operation .
3. Initialize data (variable).
4. We transfer the data which is at top in data.
5. Them we increase top from top-1 .

print

1.Initialize i .

1. With the help of for loop condition ( i=top,i>-1 ,i--)
2. Print element of array
3. Return data;\
4. Them we can print which element get deleted .

### SOURCE CODE:-

### OUTPUT:-

## EXPERIMENT NO.-08

AIM:- Write a menu driven program to implement various operations as push, pop, display, isFull and isEmpty in a stack with the help of dynamic memory allocation.

DATE:- 24/05/2023

TOOL:- Dev C++

ALGORITHM:- 1. First create a node which contains data and a pointer (next) and a another pointer top.

2. Create a node

Initialize a pointer and assign memory with the help of malloc.

Assign data in his data field.

Next of the new is equal to NULL.

3.Push

Initialize a pointer and assign memory with the help of malloc.

Assign data in his data field. next of the new is equal to top. assign new in top.

4.Pop

First we check the linked list is empty or not.

If it is not empty we pop.

Take a pointer (P) = top;

Then increase top is equal to top's next.

Then free P.

5.Print

Take a pointer (ptr) = top.

Traverse the list until ptr reaches NULL.

Then print elements of list.

Increase ptr = ptr->next;

### SOURCE CODE:-

//STACK BY LINKED LIST

#include<stdio.h> #include<stdlib.h> struct node{

int data; struct node \* next;

}\*top; typedef struct node node; node\*createnode(int data){ node\*new=(node\*)malloc(sizeof(node)); new->data=data; new->next=NULL; return new;

}

node\*push(node\*top,int data){ node\*new=(node\*)malloc(sizeof(node)); new->data=data; new->next=top; top=new; return top;

}

node\*pop(node\*top){

if(isempty()){ return 0;

}

node\*p=top;

top=top->next; free(p); return top;

}

int isempty(){

if(top==NULL){ printf("Stack is Underflow\n");

}

else{ return 0;

}

}

void printlist(node\*top)

{

node\*ptr=top;

while(ptr!=NULL){ printf("\n%d",ptr->data);

ptr=ptr->next;

}

}

int main()

{

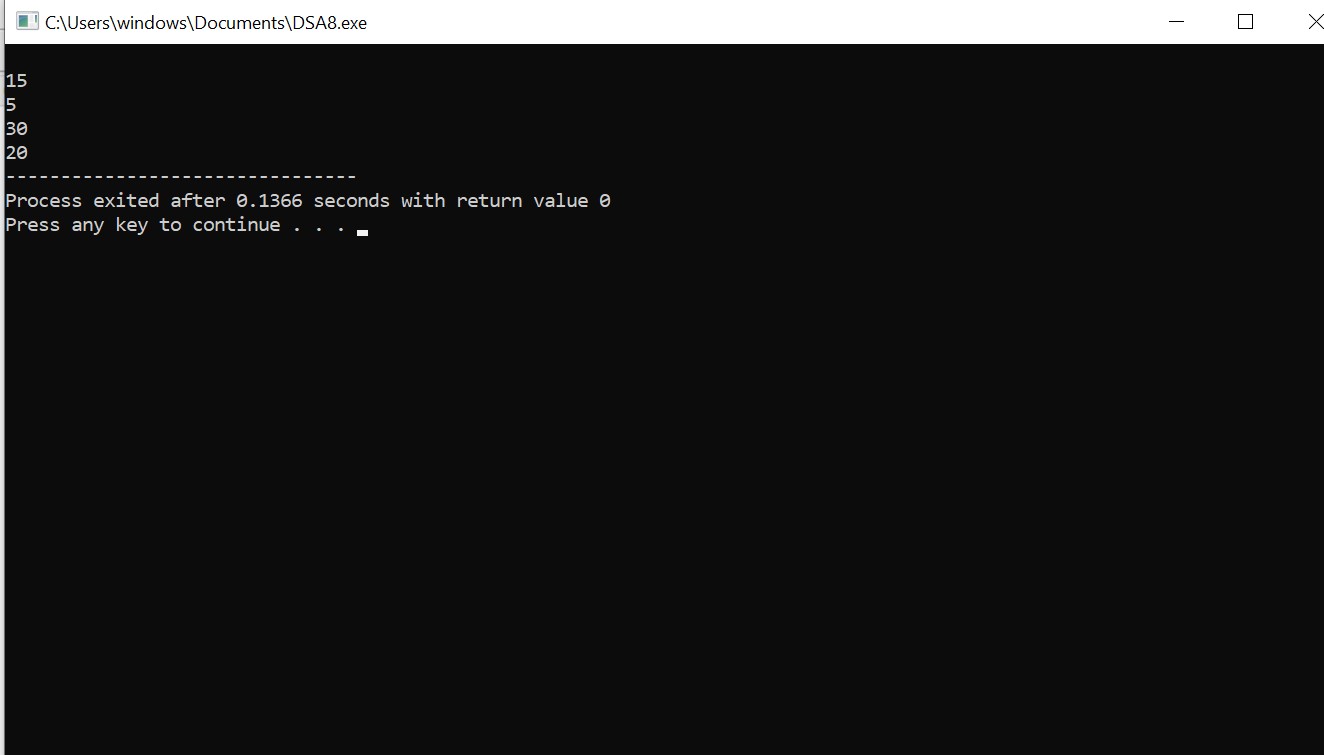
top=createnode(20); top=push(top,30); top=push(top,5); top=push(top,15); top=push(top,8); top=pop(top);

printlist(top);

return 0;

}

#### OUTPUT:-



## EXPERIMENT NO.-09

AIM:- WAP for Tower of Hanoi using recursion.

DATE:- 31/05/2023

TOOL:-Dev C++

### ALGORITHM:-

1. Take a integer n
2. get number of disk from user and scan 1 value of n
3. then we call function toh(tower of hanoi)
4. the function takes three parameters: the begining,the auxiliary, the end

The function will recursively move the disk from beg to end

1. if n==1

Then move disk 1 from beg to end

1. if(n-1)

Then move disk from beg to end

1. if(n-1)

Then move disk from aux to end

1. the function will continuously recursive untill number of disk not reaches n=1

#### SOURCE CODE:-

//Tower of hanoi #include<stdio.h> void toh(int n, char beg, char aux, char end){ if(n==1) { printf("Move disk 1 from %c to %c\n",beg,end);

return ;

}

toh(n-1,beg,end,aux); { printf("Move disk %d from %c to %c\n",n,beg,end); toh(n-1,aux,beg,end);

}

}

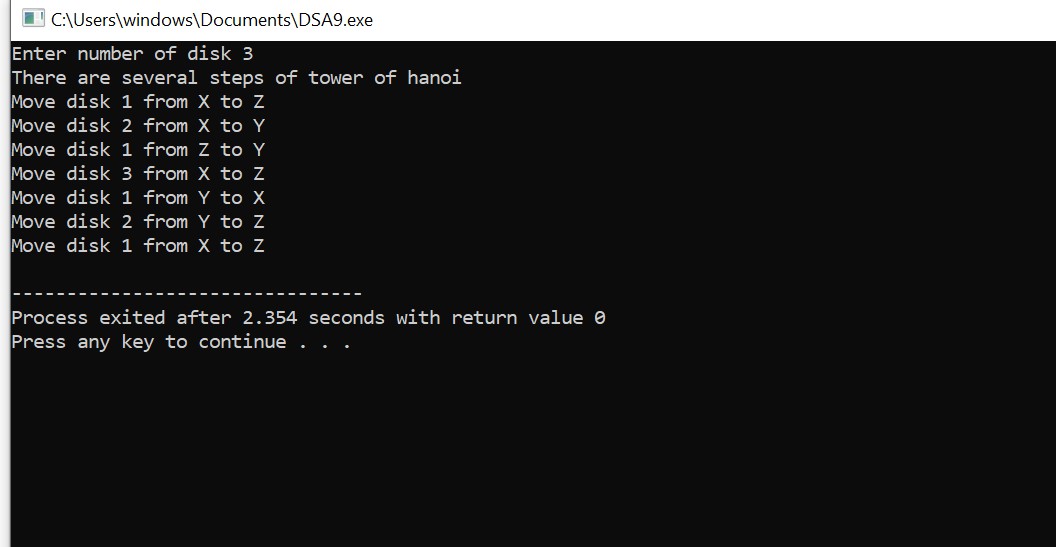
int main()

{ int n;

printf("Enter number of disk"); scanf("%d",&n); printf("There are several steps of tower of hanoi\n"); toh(n,'X','Y','Z'); return 0;

}

#### OUTPUT:-



## EXPERIMENT NO.-10

AIM:- Illustrate queue implementation using array with following operation as enQueue, deQueue, isEmpty, displayQueue.

DATE:- 07/06/2023

TOOL:-Dev C++

ALGORITHM:- 1) Define size of array.

1. Global declaration of rear = -1 , front = -1 and array ( queue [size])
2. enqueue
   * 1. if(rear == max-1)

Print queue is overflow

* + 1. else { if(rear == -1)

2.1) then increase rear and front from -1 to 0 and insert data at 0 index 2.2) else{

* 1. increase rear
  2. insert data at rear

1. dequeue
   * 1. if queue is empty then we exit directly
     2. else :- 1) int data
        1. get value of index front in data
        2. then increase front

1. print
2. Run a for loop from i = front to i <= rear and i++
3. and print element of queue

### SOURCE CODE:-

//Queue by array

#include<stdio.h> #define max 5 int rear=-1; int front=-1; int queue[max]; int enqueue(int data) { if(rear==max-1) { printf("Queue is overflow\n"); return 0;

} else { if (rear==-1) { rear=front=0; queue[0]=data;

} else { rear++; queue[rear] =data;

}

}

}

int dequeue() { if(isempty()) { exit(1); } else { int data; data=queue[front]; front++; return data;

}

}

int isempty() { if(front==-1 ) { printf("Queue is underflow\n"); return 1;

} else { return 0;

}

}

int display() { if(isempty()) { exit(1); }

else {

int i;

for(i=front; i<=rear; i++) { printf("%d\n",queue[i]);

}

}

}

int main() { int value; enqueue(10); enqueue(20); enqueue(30); enqueue(40); value=dequeue();

display(); printf("The deleted data is %d",value); return 0;

}

### OUTPUT:-

## EXPERIMENT NO.-11

AIM:- Illustrate queue implementation using linked list with following operation as enQueue, deQueue, isEmpty, displayQueue.

DATE:- 07/06/2023

TOOL:-Dev C++

### ALGORITHM:-

1. declare a structure which contains data field , next , front = NULL , rear = NULL
2. create first node
3. create a new (pointer) and assign memory from malloc function
4. assign data in his data field
5. next of new is equal to NULL
6. assign front and rear to new

1. enqueue:
   * 1. Assign memory to pointer new by malloc function
     2. set data in his data field
     3. next of new is equal to NULL
     4. next of rear equal to new
     5. update rear
2. dequeue
   1. if queue is empty then print message of underflow
   2. take pointer variable temp = front
   3. then update front from front to front next
   4. free (temp)

1. display
   1. take pointer variable, pointer start from front
   2. traverse the list from front to rear
   3. print element of first

#### SOURCE CODE:-

//Queue by linked list

#include<stdio.h> #include<stdlib.h> struct node { int data; struct node\*next;

}\*front=NULL,\*rear=NULL; typedef struct node node; node\*increate\_first\_node(int data) { node\*new=(node\*)malloc(sizeof(node)); new->data=data; new->next=NULL; front=rear=new; return front,rear;

}

node\*enqueue(int data) { node\*new=(node\*)malloc(sizeof(node)); new->data=data; new->next=NULL; rear->next=new; rear=new;

return rear;

}

node\*dequeue() {

if(isempty()) { printf("Queue is Underflow\n");

exit(1);

}

node\*temp=front; front=front->next; free(temp); return front;

}

int isempty() { if(front==NULL) { printf("Queue is underflow\n"); return 1;

} else { return 0;

}

}

int display() { if(isempty()) { exit(1);

} else { node\*ptr=front; while(ptr!=NULL) { printf("%d\n",ptr->data); ptr=ptr->next;

}

}

}

int main() { increate\_first\_node(10); enqueue(15); enqueue(20); display(); printf("After deletion\n"); dequeue(); display(); return 0;

}

#### OUTPUT:-

