Data Structure and algorithm

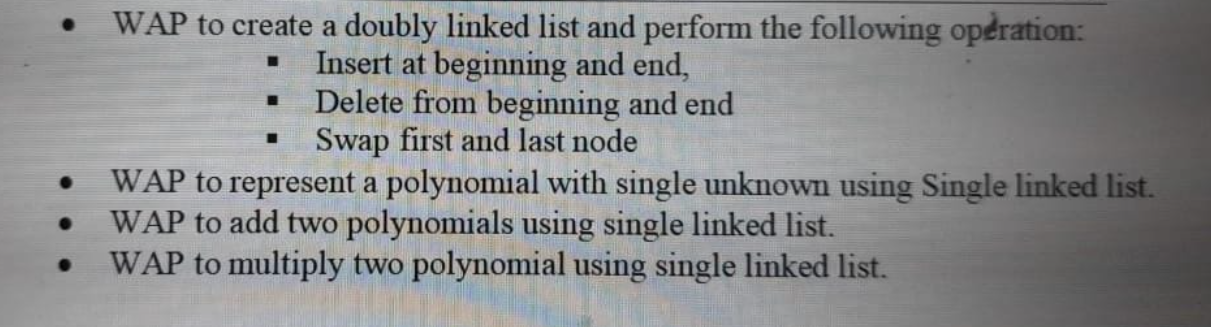
(End-sem Submission)

By:- Anusthan Singh 20051337

DSA LAB FINAL SUBMITION

Lab 8

Data:- 25/10/2021



Q1. WAP to create a doubly linked list and perform the following operation :

* Insert at beginning and end,
* Delete from beginning and end
* Swap first and last node

Code:-

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

    struct node \*prev;

    int data;

    struct node \*next;

}\*head=NULL;

void create(int d)

{

    struct node \*curr,\*ptr;

    curr=malloc(sizeof(struct node));

    printf("\nENTER THE DESIRED ELEMENT %d : ",d);

    scanf("%d",&curr->data);

    curr->prev=NULL;

    curr->next=NULL;

    if(head==NULL)

    {

        head=curr;

        ptr=curr;

    }

    else

    {

        ptr->next=curr;

        ptr=curr;

    }

}

void traverse()

{

    struct node \*cur;

    cur=head;

    printf("\n\nTHE NODE IS : ");

    while(cur!=NULL)

    {

        printf("%d ",cur->data);

        cur=cur->next;

    }

}

void insert\_begin()

{

    struct node \*cur;

    cur=malloc(sizeof(struct node));

    printf("\nENTER ELEMENT : ");

    scanf("%d",&cur->data);

    cur->prev=NULL;

    cur->next=NULL;

    if(head==NULL)

    {

        cur=head;

    }

    else

    {

       cur->next=head;

       head=cur;

    }

}

void insert\_end()

{

    struct node \*cur,\*ptr;

    cur=head;

    while(cur->next!=NULL)

    {

        cur=cur->next;

    }

    ptr=malloc(sizeof(struct node));

    printf("\nENTER ELEMENT : ");

    scanf("%d",&ptr->data);

    ptr->prev=NULL;

    ptr->next=NULL;

    if(head==NULL)

    {

        cur=head;

    }

    else

    {

       cur->next=ptr;

       ptr->next=NULL;

       ptr->prev=cur;

    }

}

void deletbegin()

{

    struct node \*cur;

    cur=head;

    if(cur==NULL)

    {

        printf("\nNOT POSSIBLE ");

    }

    else

    {

        head=head->next;

        free(cur);

    }

}

void deletend()

{

    struct node \*cur,\*ptr;

    cur=head;

    while(cur->next!=NULL)

    {

        ptr=cur;

        cur=cur->next;

    }

    if(head==NULL)

    {

        printf("\nDELETION OF NODE NOT POSSIBLE ");

    }

    else

    {

        ptr->next=NULL;

        free(cur);

    }

}

void swap(int no)

{

    struct node \*cur;

    cur=head;

    if(head==NULL)

    {

        printf("\nTHE SWAP IS NOT POSSIBLE ");

    }

    else if(no!=1)

    {

       struct node \*cur,\*ptr;

       cur=head;

       int t;

       while(cur->next!=NULL)

       {

           cur=cur->next;

       }

       ptr=head;

       t=ptr->data;

       ptr->data=cur->data;

       cur->data=t;

    }

}

void main()

{

    struct node \*cur,\*ptr;

    int n,i,ch,d=1;

    printf("ENTER THE NUMBER OF ELEMENTS YOU WANT TO ENTER : ");

    scanf("%d",&n);

    for(i=0;i<n;i++)

    {

       create(i+1);

    }

   while(d!=0)

   {

   printf("\n 1. TRAVERSE");

   printf("\n 2.  INSERT AT BEGINNING");

   printf("\n 3.  INSERT AT END");

   printf("\n 4.  SWAP");

   printf("\n 5.  DELETE AT BEGINNING");

   printf("\n 6.  DELETE AT END");

   printf("\nENTER YOUR CHOICE : ");

   scanf("%d",&ch);

   printf("\n");

   switch(ch)

   {

      case 1:

      {

          traverse();

          break;

      }

      case 2:

      {

          insert\_begin();

          traverse();

          break;

      }

      case 3:

      {

          insert\_end();

          traverse();

          break;

      }

      case 4:

      {

          swap(n);

          traverse();

          break;

      }

      case 5:

      {

          deletbegin();

          traverse();

          break;

      }

      case 6:

      {

          deletend();

          traverse();

          break;

      }

      default:

     {

        printf("\n INVALID CHOICE ");

        break;

     }

    }

        printf("\n");

        printf("\nWANT TO RUN THE LOOP : ");

        scanf("%d",&d);

    }

   getch();

}

Output :

ENTER THE NUMBER OF ELEMENTS YOU WANT TO ENTER : 4

ENTER THE DESIRED ELEMENT 1 : 1

ENTER THE DESIRED ELEMENT 2 : 2

ENTER THE DESIRED ELEMENT 3 : 3

ENTER THE DESIRED ELEMENT 4 : 4

1. TRAVERSE

2. INSERT AT BEGINNING

3. INSERT AT END

4. SWAP

5. DELETE AT BEGINNING

6. DELETE AT END

ENTER YOUR CHOICE : 2

ENTER ELEMENT : 5

THE NODE IS : 5 1 2 3 4

WANT TO RUN THE LOOP : 3

1. TRAVERSE

2. INSERT AT BEGINNING

3. INSERT AT END

4. SWAP

5. DELETE AT BEGINNING

6. DELETE AT END

ENTER YOUR CHOICE : 3

ENTER ELEMENT : 4

THE NODE IS : 5 1 2 3 4 4

WANT TO RUN THE LOOP : 5

1. TRAVERSE

2. INSERT AT BEGINNING

3. INSERT AT END

4. SWAP

5. DELETE AT BEGINNING

6. DELETE AT END

ENTER YOUR CHOICE : 5

THE NODE IS : 1 2 3 4 4

WANT TO RUN THE LOOP : 6

1. TRAVERSE

2. INSERT AT BEGINNING

3. INSERT AT END

4. SWAP

5. DELETE AT BEGINNING

6. DELETE AT END

ENTER YOUR CHOICE : 6

THE NODE IS : 1 2 3 4

WANT TO RUN THE LOOP : 4

1. TRAVERSE

2. INSERT AT BEGINNING

3. INSERT AT END

4. SWAP

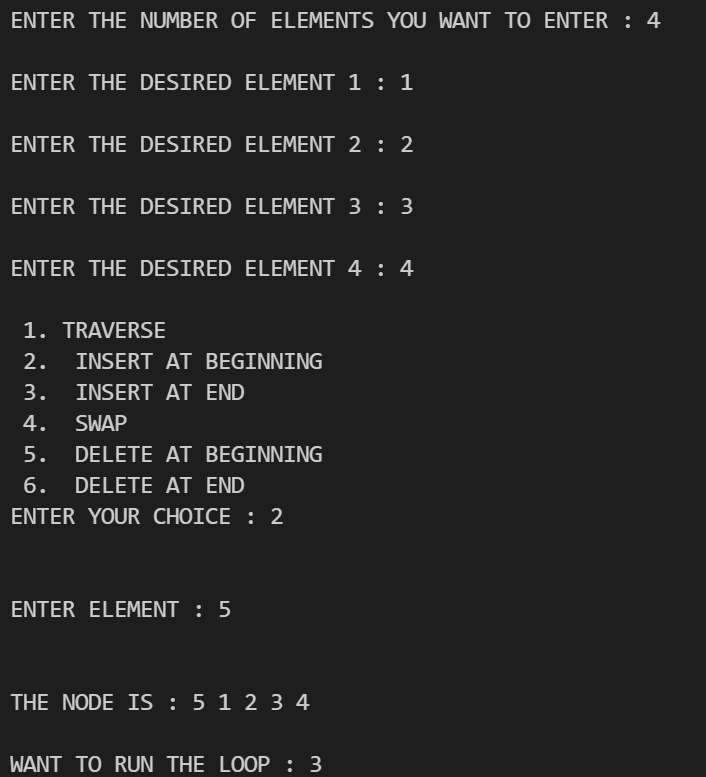
5. DELETE AT BEGINNING

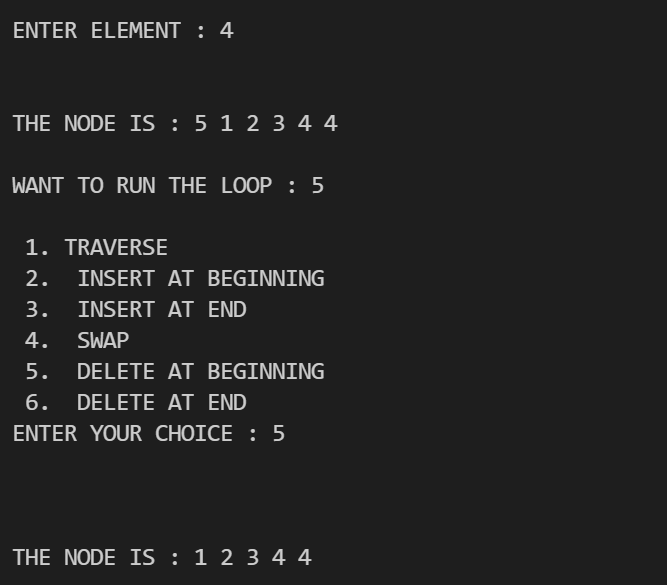
6. DELETE AT END

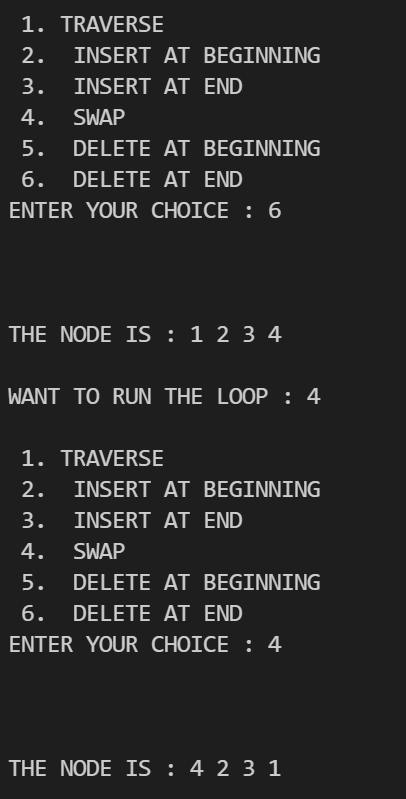
ENTER YOUR CHOICE : 4

THE NODE IS : 4 2 3 1

Screenshot:-







Q2 .WAP to represent a polynomial with single unknown using Single linked list

Code:-

#include<stdlib.h>

//decleration

struct node

{

    struct node \*prev;

    int data;

    struct node \*next;

}\*first;

//creation

void create(int a[],int n)

{

    struct node \*t,\*last;

    int i;

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

    first->data=a[0];

    first->next=first->prev=NULL;

    last=first;

    for(int i=1;i<n;i++)

    {

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

        t->data=a[i];

        t->next=last->next;

        t->prev=last;

        last->next=t;

        last=t;

    }

}

//display

void display(struct node \*p)

{

    while(p)

    {

        printf("%d ",p->data);

        p=p->next;

    }

    printf("\n");

}

//length creation

int length(struct node \*p)

{

    int len=0;

    while(p)

    {

        len++;

        p=p->next;

    }

    return len;

}

//INSERTION

void insert(struct node \*p,int index,int x)

{

    struct node \*t;

    int i;

    if(index<0|| index>length(p))

        return;

    if(index==0)

    {

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

        t->data=x;

        t->prev=NULL;

        t->next=first;

        first->prev=t;

        first=t;

    }

    else

    {

        for(int i=1;i<index-1;i++)

            p=p->next;

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

        t->data=x;

        t->next=p->next;

        t->prev=p;

        if(p->next)

        {

            p->next->prev=t;

        }

        p->next=t;

    }

}

//DELETION

int Delete(struct node \*p,int index)

{

    struct node \*q;

    int x=-1,i;

    if(index<0||index>length(p))

        return x;

    if(index==1)

    {

        first=first->next;

        if(first)

        {

            first->prev=NULL;

        }

        x=p->data;

        free(p);

    }

    else

    {

        for(int i=1;i<index-1;i++)

        p=p->next;

        p->prev->next=p->next;

        if(p->next)

        {

            p->next->prev=p->prev;

        }

        x=p->data;

        free(p);

    }

}

int main()

{

    int a[]={4,5,6,7,8};

    create(a,5);

    printf("length of linked list is :%d \n",length(first));

    insert(first,5,90);

    Delete(first,4);

    display(first);

}

Output :

length of linked list is :5

4 5 7 90 8

Screenshot:-



Q3.WAP to add two polynomials using single linked list.

Code : #include<stdio.h>

#include<stdlib.h>

struct node

{

    int coef,expo;

    struct node\* next;

};

struct node\* insertion(struct node\* thead,int c,int e);

struct node\* attendent(struct node\* thead,int c,int e);

struct node\* add(struct node\* p1thead,struct node\* p2thead);

void display(struct node\* thead);

void main()

{

    int z,y,n,i;

    struct node\* p1head,\* p2head,\* p3head;

    p1head=p2head=NULL;

//1st intake

    printf("Enter the no of terms of 1st polynomial ");

    scanf("%d",&n);

    printf("\nEnter the polynomial = \n");

    for(i=0;i<n;i++){

        printf("\nEnter the coefficient and exponent of the term =");

        scanf("%d%d",&z,&y);

        p1head=insertion(p1head,z,y);

    }

//2nd intake

    printf("\nEnter the no of terms of 2nd polynomial  ");

    scanf("%d",&n);

    printf("\nEnter the polynomial =\n ");

    for(i=0;i<n;i++){

        printf("\nEnter the coefficient and exponent of the term");

        scanf("%d%d",&z,&y);

        p2head=insertion(p2head,z,y);

    }

    //Performing Addition

    p3head=add(p1head,p2head);

    //Displaying

    printf("\nThe polynomial 1 is");

    display(p1head);

    printf("\nThe polynomial 2 is");

    display(p2head);

    printf("\nThe sum of the two polynomials is");

    display(p3head);

}

struct node\* attendent(struct node\* thead,int c,int e)

{

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

    newnode->coef=c;

    newnode->expo=e;

    if(thead==NULL){ // Corner Case to handle if the list is empty...

    newnode->next=NULL;

    return newnode;

    }

    struct node\* trav=thead;

    while(trav->next!=NULL)  // Traversing to point to the last node...

        trav=trav->next;

    trav->next=newnode;

    newnode->next=NULL;

    return thead;

}

struct node\* insertion(struct node\* thead,int c,int e)

{

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

    newnode->coef=c;

    newnode->expo=e;

    if(thead==NULL){            // for inserting the first node..

        newnode->next=NULL;

        return newnode;

    }

    struct node\* prev,\* curr;

    prev=curr=thead;

    while(curr!=NULL && curr->expo>e){

        prev=curr;

        curr=curr->next;

    }

    if(curr==thead){            // for inserting before the first node

        newnode->next=curr;

        return newnode;

    }

    else if(curr==NULL){        //for inserting after the last node

        prev->next=newnode;

        newnode->next=NULL;

    }

    else{

        newnode->next=curr;

        prev->next=newnode;

    }

    return thead;

}

struct node\* add(struct node\* p1thead,struct node\* p2thead)

{

    struct node\* ans=NULL;

    struct node\* t1,\* t2;

    t1=p1thead;

    t2=p2thead;

    while(t1!=NULL && t2!=NULL){

        if(t1->expo > t2->expo){

            ans=attendent(ans,t1->coef,t1->expo);

            t1=t1->next;

        }

        else if(t1->expo < t2->expo){

            ans=attendent(ans,t2->coef,t2->expo);

            t2=t2->next;

        }

        else{

            ans=attendent(ans,(t1->coef)+(t2->coef),t1->expo);

            t1=t1->next;

            t2=t2->next;

        }

    }

    while(t1!=NULL){            //coping the remaining terms of polynomial 1

        ans=attendent(ans,t1->coef,t1->expo);

        t1=t1->next;

    }

    while(t2!=NULL){            //coping the remaining terms of polynomial 2

        ans=attendent(ans,t2->coef,t2->expo);

        t2=t2->next;

    }

    return ans;

}

void display(struct node\* thead)

{

    struct node\* temp=thead;

    if(temp==NULL){

        printf("\nEmpty..");

    }

    else{

        while(temp->next!=NULL){

            printf(" %dx^%d +",temp->coef,temp->expo);

            temp=temp->next;

        }

       printf(" %dx^%d ",temp->coef,temp->expo);

    }

}

Output :

Enter the no of terms of 1st polynomial 2

Enter the polynomial =

Enter the coefficient and exponent of the term =1

2

Enter the coefficient and exponent of the term =3

4

Enter the no of terms of 2nd polynomial 2

Enter the polynomial =

Enter the coefficient and exponent of the term1

2

Enter the coefficient and exponent of the term3

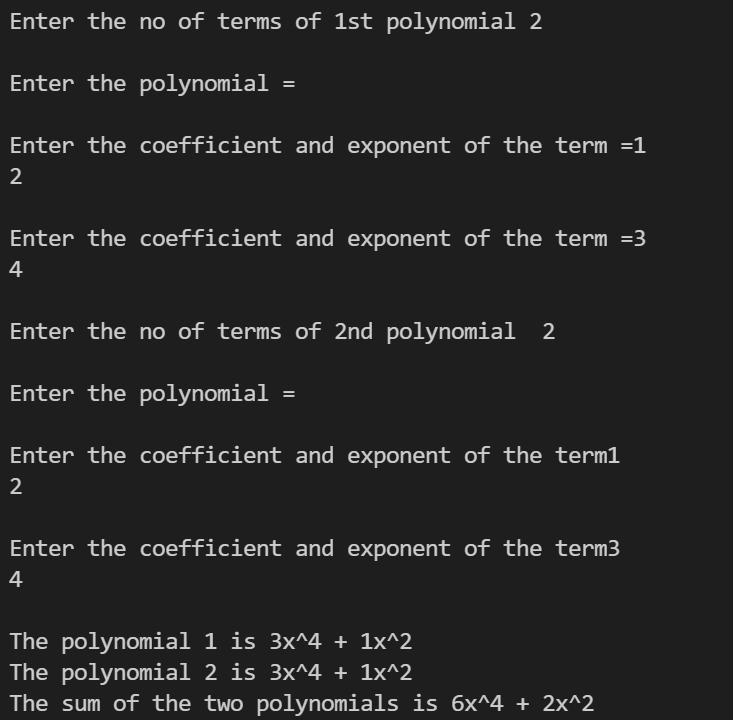
4

The polynomial 1 is 3x^4 + 1x^2

The polynomial 2 is 3x^4 + 1x^2

The sum of the two polynomials is 6x^4 + 2x^2

Screenshot:



Q4.WAP to multiply two polynomial using single linked list.

Code:-

 #include<stdio.h>

#include<stdlib.h>

struct node

{

 int coefficient,exponent;

 struct node \*next;

};

struct node \*hPtr1,\*hPtr2,\*hPtr3;

struct node \*buildNode(int coefficient, int exponent)

{

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

 ptr->coefficient=coefficient;

 ptr->exponent=exponent;

 ptr->next=NULL;

 return ptr;

}

void polynomial\_insert(struct node \*\* myNode,int coefficient,int exponent)

{

 struct node \*lPtr,\*pPtr,\*qPtr=\*myNode;

 lPtr=buildNode(coefficient,exponent);

 if (\*myNode==NULL || (\*myNode)->exponent<exponent)

 {

  \*myNode=lPtr;

  (\*myNode)->next=qPtr;

  return;

 }

 while(qPtr)

 {

  pPtr=qPtr;

  qPtr=qPtr->next;

  if(!qPtr)

  {

   pPtr->next=lPtr;

   break;

  }

  else if((exponent<pPtr->exponent) && (exponent>qPtr->exponent))

  {

   lPtr->next = qPtr;

   pPtr->next = lPtr;

   break;

  }

 }

 return;

}

void polynomial\_add(struct node \*\*n1,int coefficient,int exponent)

{

 struct node \*x=NULL,\*temp=\*n1;

 if (\*n1==NULL || (\*n1)->exponent<exponent)

 {

  \*n1=x=buildNode(coefficient,exponent);

  (\*n1)->next=temp;

 }

 else

 {

  while(temp)

  {

   if(temp->exponent==exponent)

   {

    temp->coefficient=temp->coefficient+coefficient;

    return;

   }

   if(temp->exponent>exponent && (!temp->next || temp->next->exponent<exponent))

   {

    x=buildNode(coefficient,exponent);

    x->next=temp->next;

    temp->next=x;

    return;

   }

   temp=temp->next;

  }

  x->next=NULL;

  temp->next=x;

 }

}

void polynomial\_multiply(struct node \*\*n1,struct node \*n2,struct node \*n3)

{

 struct node \* temp;

 int coefficient, exponent;

 temp = n3;

 if(!n2 && !n3)

 {

  return;

 }

 if(!n2)

 {

  \*n1 = n3;

 }

 else if(!n3)

 {

  \*n1 = n2;

 }

 else

 {

  while(n2)

  {

   while(n3)

   {

    coefficient = n2->coefficient \* n3->coefficient;

    exponent = n2->exponent + n3->exponent;

    n3 = n3->next;

    polynomial\_add(n1, coefficient, exponent);

   }

   n3 = temp;

   n2 = n2->next;

  }

 }

 return;

}

struct node \*polynomial\_deleteList(struct node \*ptr)

{

 struct node \*temp;

 while(ptr)

 {

  temp=ptr->next;

  free(ptr);

  ptr = temp;

 }

 return NULL;

}

void polynomial\_view(struct node \*ptr)

{

 int i = 0;

 int flag=0;

 while (ptr)

 {

  if(ptr->exponent!=0 || ptr->exponent!= 1)

  {

   if(ptr->coefficient>0 && flag==0)

   {

    printf("%dx^%d", ptr->coefficient,ptr->exponent);

    flag++;

   }

   else if(ptr->coefficient>0 && flag==1)

   {

    printf("+%dx^%d", ptr->coefficient,ptr->exponent);

   }

   else if(ptr->coefficient<0)

   {

    printf("%dx^%d",ptr->coefficient,ptr->exponent);

   }

  }

  else if(ptr->exponent==0)

  {

   if(ptr->coefficient>0 && flag==0 )

   {

    printf("%d",ptr->coefficient);

      flag++;

   }

   else if(ptr->coefficient>0 && flag==1)

   {

    printf("+%d", ptr->coefficient);

   }

   else if(ptr->coefficient < 0)

   {

    printf("%d", ptr->coefficient);

   }

  }

  else if(ptr->exponent==1)

  {

   if(ptr->coefficient>0 && flag==0)

   {

    printf("%dx",ptr->coefficient);

    flag++;

   }

   else if(ptr->coefficient > 0 && flag==1)

   {

    printf("+%dx", ptr->coefficient);

   }

   else if(ptr->coefficient < 0)

   {

    printf("%dx", ptr->coefficient);

   }

  }

  ptr=ptr->next;

  i++;

 }

 printf("\n");

 return;

}

int main(int argc,char \*argv[])

{

 int coefficient,exponent,i,n;

 int count;

 printf("Multiplication of Two Polynomials\n");

 printf("Enter the number of coefficients in the multiplicand:");

 scanf("%d",&count);

 for(i=0;i<count;i++)

 {

  printf("Enter the coefficient part:");

  scanf("%d", &coefficient);

  printf("Enter the exponent part:");

  scanf("%d",&exponent);

  polynomial\_insert(&hPtr1, coefficient, exponent);

 }

 printf("Enter the number of coefficients in the multiplier:");

 scanf("%d",&count);

 for(i=0;i<count;i++)

 {

  printf("Enter the coefficient part:");

  scanf("%d", &coefficient);

  printf("Enter the exponent part:");

  scanf("%d",&exponent);

  polynomial\_insert(&hPtr2, coefficient, exponent);

 }

 printf("Polynomial Expression 1: ");

 polynomial\_view(hPtr1);

 printf("Polynomial Expression 2: ");

 polynomial\_view(hPtr2);

 polynomial\_multiply(&hPtr3, hPtr1, hPtr2);

 printf("Output:\n");

 polynomial\_view(hPtr3);

 hPtr1 = polynomial\_deleteList(hPtr1);

 hPtr2 = polynomial\_deleteList(hPtr2);

 hPtr3 = polynomial\_deleteList(hPtr3);

 return 0;

}

Output:-

Multiplication of Two Polynomials

Enter the number of coefficients in the multiplicand:2

Enter the coefficient part:2

Enter the exponent part:2

Enter the coefficient part:3

Enter the exponent part:3

Enter the number of coefficients in the multiplier:2

Enter the coefficient part:2

Enter the exponent part:2

Enter the coefficient part:3

Enter the exponent part:3

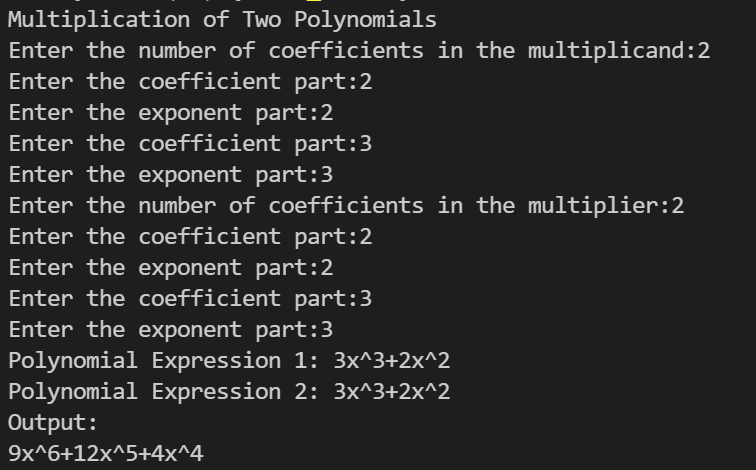
Polynomial Expression 1: 3x^3+2x^2

Polynomial Expression 2: 3x^3+2x^2

Output:

9x^6+12x^5+4x^4

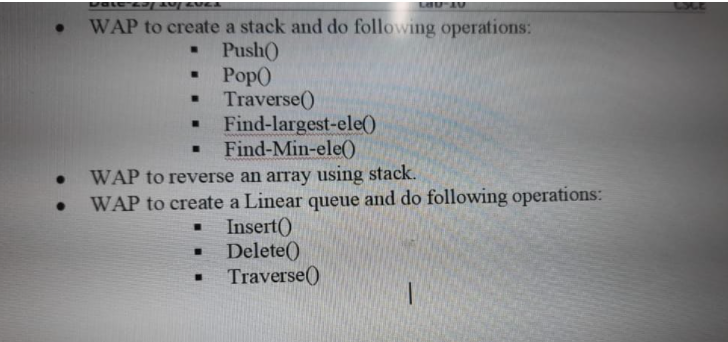
Screenshot:-



Lab 9

Date:- 1/11/2021

Q



Q1. WAP to create a stack and do following operations:

: ■ Push0

▪ POP()

■ Traverse()

■ Find-largest-ele()

■ Find7Min-ele()

Code:-

#include<stdio.h>

#include<stdlib.h>

#include<limits.h>

struct stack

{

    unsigned size;

    int \*arr,top;

};

struct stack\* stackcreation(unsigned size)

{

    struct stack\* stack = malloc(sizeof(struct stack));

    stack->size=size;

    stack->top=-1;

    stack->arr=malloc(sizeof(int)\*stack->size);

    return stack;

}

void push(struct stack\* stack, int elements)

{

    if(stack->top==stack->size-1) return;

    stack->arr[++(stack->top)] = elements;

}

int pop(struct stack\* stack)

{

    if(stack->top<0) return INT\_MIN;

    return stack->arr[stack->top--];

}

void display(struct stack\* stack)

{

    for(int i=stack->top;i>=0;i--)

    printf("%d ",stack->arr[i]);

}

int largest\_ele (struct stack\* stack)

{

    int i,largest=stack->arr[0];

    for(int i=stack->top;i>=0;i--)

    if(stack->arr[i]>largest)

    largest=stack->arr[i];

    return largest;

}

int smallest\_ele (struct stack\* stack)

{

    int i,temp=stack->arr[1];

    for(int i=stack->top;i>=0;i--)

    if(stack->arr[i]<temp)

    temp=stack->arr[i];

    return temp;

}

int main()

{

    printf("\*\*\*\*Stack creation push pull traverse\*\*\*\*\*\n");

    printf("Elements : ");

    struct stack\* stack = stackcreation(5);

    for(int i=1;i<=4;i++) push(stack,i);

    display(stack);

    printf("\nLargest element in stack : %d",largest\_ele(stack));

    printf("\nSmallest element in stack : %d",smallest\_ele(stack));

    printf("\n%d",pop(stack));

    printf("\n%d\n",pop(stack));

    display(stack);

    printf("\n\*\*\*\*Program ended successfully\*\*\*\*\*");

    return 0;

}

Output:-

\*\*\*\*Stack creation push pull traverse\*\*\*\*\*

Elements : 4 3 2 1

Largest element in stack : 4

Smallest element in stack : 1

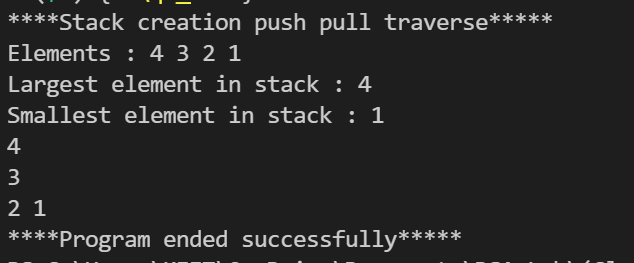
4

3

2 1

\*\*\*\*Program ended successfully\*\*\*\*\*

Screenshot:-



Q2.

• WAP to reverse an array using stack.

Code:-

#include <stdio.h>

#include <stdlib.h>

struct Stack

{

    int \*arr;

    int top;

    unsigned space;

};

struct Stack \*create(unsigned capacity)

{

    struct Stack \*stack = (struct Stack \*)malloc(sizeof(struct Stack));

    stack->space = capacity;

    stack->top = -1;

    stack->arr = (int \*)malloc(stack->space \* sizeof(int));

    return stack;

}

int isFull(struct Stack \*stack)

{

    return stack->top == stack->space - 1;

}

int isEmpty(struct Stack \*stack)

{

    return stack->top == -1;

}

void push(struct Stack \*stack, int item)

{

    if (isFull(stack))

        return;

    stack->arr[++stack->top] = item;

}

int pop(struct Stack \*stack)

{

    if (isEmpty(stack))

        return -1;

    return stack->arr[stack->top--];

}

void reverseArrayS(int arr[], int n)

{

    struct Stack \*stack = create(n);

    for (int i = 0; i < n; i++)

    {

        push(stack, arr[i]);

    }

    for (int i = 0; i < n; i++)

    {

        arr[i] = pop(stack);

    }

    for (int i = 0; i < n; i++)

        printf("%d ", arr[i]);

}

int main()

{

    int a=0,b=0,c=0,d=0;

    printf("Enter value of of A,B,C,D ");

    scanf("%d",&a);

      scanf("%d",&b);

        scanf("%d",&c);

          scanf("%d",&d);

    int arr[] = {a,b,c,d};

    int N = sizeof(arr) / sizeof(arr[0]);

    reverseArrayS(arr, N);

    return 0;

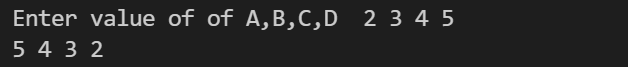
}

Output:-

Enter value of of A,B,C,D 2 3 4 5

5 4 3 2

Screenshoot:-



Q3.

• WAP to create a Linear queue and do following operations:

■ Insert()

■ Delete()

■ Traverse()

Code:-

#include<stdlib.h>

struct node

{

    struct node \*prev;

    int data;

    struct node \*next;

}\*first;

void creation(int a[],int n)

{

    struct node \*t,\*last;

    int i;

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

    first->data=a[0];

    first->next=first->prev=NULL;

    last=first;

    for(int i=1;i<n;i++)

    {

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

        t->data=a[i];

        t->next=last->next;

        t->prev=last;

        last->next=t;

        last=t;

    }

}

void display(struct node \*p)

{

    while(p)

    {

        printf("%d ",p->data);

        p=p->next;

    }

    printf("\n");

}

//length no need

int length(struct node \*p)

{

    int len=0;

    while(p)

    {

        len++;

        p=p->next;

    }

    return len;

}

void insertion(struct node \*p,int index,int x)

{

    struct node \*t;

    int i;

    if(index<0|| index>length(p))

        return;

    if(index==0)

    {

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

        t->data=x;

        t->prev=NULL;

        t->next=first;

        first->prev=t;

        first=t;

    }

    else

    {

        for(int i=1;i<index-1;i++)

            p=p->next;

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

        t->data=x;

        t->next=p->next;

        t->prev=p;

        if(p->next)

        {

            p->next->prev=t;

        }

        p->next=t;

    }

}

int Deletion(struct node \*p,int index)

{

    struct node \*q;

    int x=-1,i;

    if(index<0||index>length(p))

        return x;

    if(index==1)

    {

        first=first->next;

        if(first)

        {

            first->prev=NULL;

        }

        x=p->data;

        free(p);

    }

    else

    {

        for(int i=1;i<index-1;i++)

        p=p->next;

        p->prev->next=p->next;

        if(p->next)

        {

            p->next->prev=p->prev;

        }

        x=p->data;

        free(p);

    }

}

int main()

{

    int a[]={1,2,3,4,5};

    creation(a,5);

    insertion(first,5,15);

    Deletion(first,4);

    display(first);

}

Output:-

1 2 4 15 5

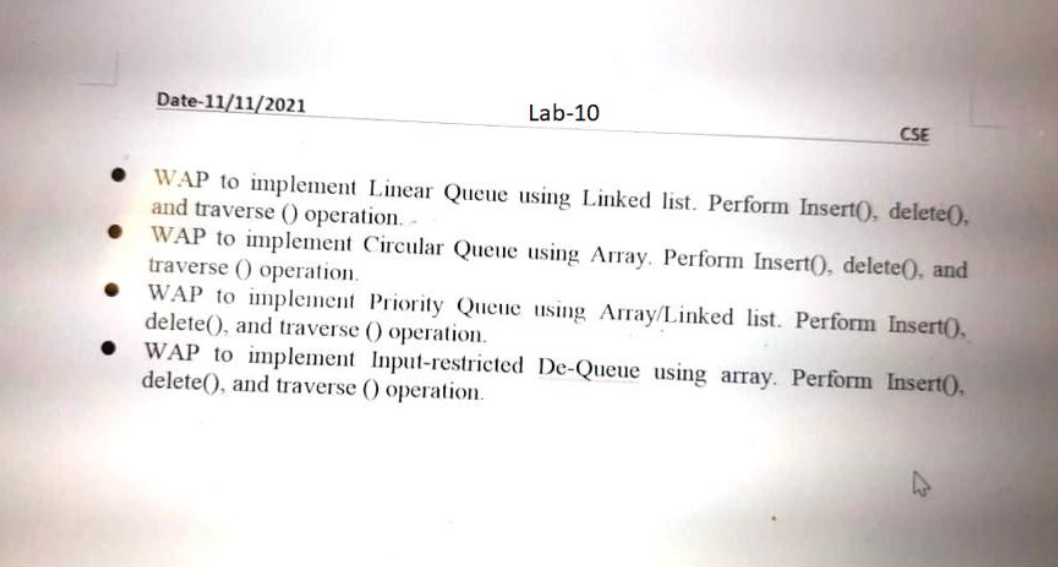
Screenshot:-



Lab 10

Date:- 8/11/2021

Q



Q1. • WAP to implement Linear Queue using Linked list. Perform Insert°. delete(), and traverse () operation.

Code:-

#include<stdlib.h>

struct node

{

    struct node \*prev;

    int data;

    struct node \*next;

}\*first;

void creation(int a[],int n)

{

    struct node \*t,\*last;

    int i;

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

    first->data=a[0];

    first->next=first->prev=NULL;

    last=first;

    for(int i=1;i<n;i++)

    {

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

        t->data=a[i];

        t->next=last->next;

        t->prev=last;

        last->next=t;

        last=t;

    }

}

void display(struct node \*p)

{

    while(p)

    {

        printf("%d ",p->data);

        p=p->next;

    }

    printf("\n");

}

//length no need

int length(struct node \*p)

{

    int len=0;

    while(p)

    {

        len++;

        p=p->next;

    }

    return len;

}

void insertion(struct node \*p,int index,int x)

{

    struct node \*t;

    int i;

    if(index<0|| index>length(p))

        return;

    if(index==0)

    {

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

        t->data=x;

        t->prev=NULL;

        t->next=first;

        first->prev=t;

        first=t;

    }

    else

    {

        for(int i=1;i<index-1;i++)

            p=p->next;

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

        t->data=x;

        t->next=p->next;

        t->prev=p;

        if(p->next)

        {

            p->next->prev=t;

        }

        p->next=t;

    }

}

int Deletion(struct node \*p,int index)

{

    struct node \*q;

    int x=-1,i;

    if(index<0||index>length(p))

        return x;

    if(index==1)

    {

        first=first->next;

        if(first)

        {

            first->prev=NULL;

        }

        x=p->data;

        free(p);

    }

    else

    {

        for(int i=1;i<index-1;i++)

        p=p->next;

        p->prev->next=p->next;

        if(p->next)

        {

            p->next->prev=p->prev;

        }

        x=p->data;

        free(p);

    }

}

int main()

{

    int a[]={1,2,3,4,5,6,7,8,9};

    creation(a,9);

    //printf("length of linked list is :%d \n",length(first));

    insertion(first,5,15);

    Deletion(first,4);

    display(first);

}

Output:-

1 2 4 15 5 6 7 8 9

Screenshot:-



Q2. • WAP to implement Circular Queue using Array. Perform Insert°. delete°. and traverse 0 operation:

# include<stdio.h>

# define MAX 5

int cqueue\_arr[MAX];

int front = -1;

int rear = -1;

void insert(int item)

{

    if((front == 0 && rear == MAX-1) || (front == rear+1))

    {

        printf("Queue Overflow \n");

        return;

    }

    if (front == -1)  // or If queue  empty

    {

        front = 0;

        rear = 0;

    }

    else

    {

        if(rear == MAX-1)

            rear = 0;

        else

            rear = rear+1;

    }

    cqueue\_arr[rear] = item ;

}

void delet()

{

    if (front == -1)

    {

        printf("Queue Underflow\n");

        return ;

    }

    printf("Element deleted from queue is : %d\n",cqueue\_arr[front]);

    if(front == rear)

    {

        front = -1;

        rear=-1;

    }

    else

    {

        if(front == MAX-1)

            front = 0;

        else

            front = front+1;

    }

}

void display()

{

    int front\_pos = front,rear\_pos = rear;

    if(front == -1)

    {

        printf("Queue is empty\n");

        return;

    }

    printf("Queue elements :\n");

    if( front\_pos <= rear\_pos )

        while(front\_pos <= rear\_pos)

        {

            printf("%d ",cqueue\_arr[front\_pos]);

            front\_pos++;

        }

    else

    {

        while(front\_pos <= MAX-1)

        {

            printf("%d ",cqueue\_arr[front\_pos]);

            front\_pos++;

        }

        front\_pos = 0;

        while(front\_pos <= rear\_pos)

        {

            printf("%d ",cqueue\_arr[front\_pos]);

            front\_pos++;

        }

    }

    printf("\n");

}

int main()

{

    int choice,item;

    do

    {

        printf("1.Insert\n");

        printf("2.Delete\n");

        printf("3.Display\n");

        printf("4.Quit\n");

        printf("Enter your choice : ");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1 :

                printf("Input the element for insertion in queue : ");

                scanf("%d", &item);

                insert(item);

                break;

            case 2 :

                delet();

                break;

            case 3:

                display();

                break;

            case 4:

                break;

                default:

                printf("Wrong choice\n");

        }

    }while(choice!=4);

    return 0;

}

Output:-

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 1

Input the element for insertion in queue : 2

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 1

Input the element for insertion in queue : 3

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 1

Input the element for insertion in queue : 4

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 1

Input the element for insertion in queue : 5

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 2

Element deleted from queue is : 2

1.Insert

2.Delete

3.Display

4.Quit

Enter your choice : 3

Queue elements :

3 4 5

1.Insert

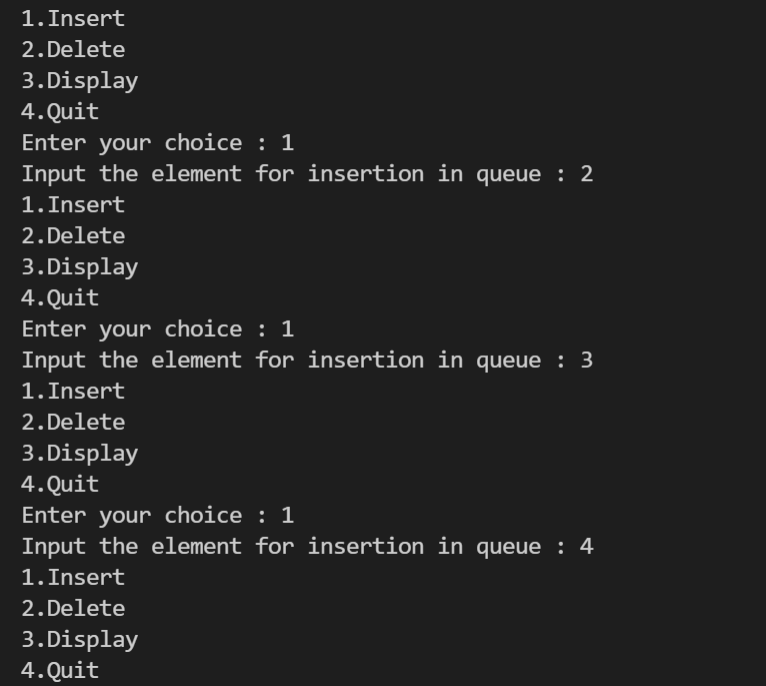
2.Delete

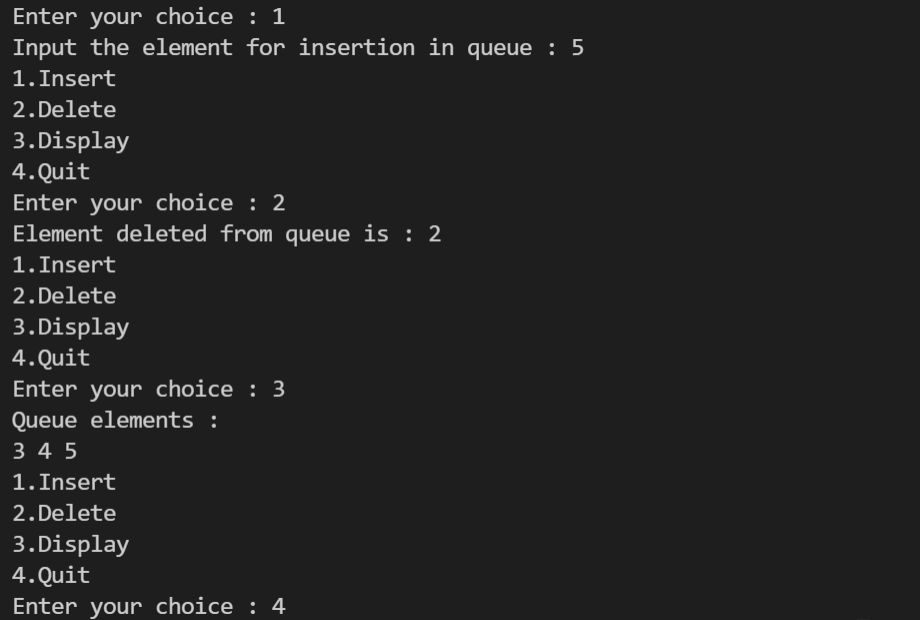
3.Display

4.Quit

Enter your choice : 4

Screenshot:-





Q3. . • WAP to implement Priority Queue using Array/Linked list. Perform Insert°,. delete(), and traverse 0 operation:

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

void insert\_by\_priority(int);

void delete\_by\_priority(int);

void create();

void check(int);

void display\_priorityqueue();

int p\_queue[MAX];

int front, rear;

void main()

{

    int n, choice;

    printf("\n1. Insert ");

    printf("\n2.  Delete ");

    printf("\n3.  Display ");

    printf("\n4.  Quit");

    create();

    while (1)

    {

        printf("\nEnter your choice : ");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            printf("\nEnter value to be inserted : ");

            scanf("%d",&n);

            insert\_by\_priority(n);

            break;

        case 2:

            printf("\nEnter value to delete : ");

            scanf("%d",&n);

            delete\_by\_priority(n);

            break;

        case 3:

            display\_priorityqueue();

            break;

        case 4:

            exit(0);

        default:

            printf("\nChoice is incorrect, Enter a correct choice");

        }

    }

}

// empty priority queue

void create()

{

    front = rear = -1;

}

void insert\_by\_priority(int data)

{

    if (rear >= MAX - 1)

    {

        printf("\nQueue overflow no more elements can be inserted");

        return;

    }

    if ((front == -1) && (rear == -1))

    {

        front++;

        rear++;

        p\_queue[rear] = data;

        return;

    }

    else

        check(data);

    rear++;

}

void check(int data)

{

    int i,j;

    for (i = 0; i <= rear; i++)

    {

        if (data >= p\_queue[i])

        {

            for (j = rear + 1; j > i; j--)

            {

                p\_queue[j] = p\_queue[j - 1];

            }

            p\_queue[i] = data;

            return;

        }

    }

    p\_queue[i] = data;

}

void delete\_by\_priority(int data)

{

    int i;

    if ((front==-1) && (rear==-1))

    {

        printf("\nQueue is empty no elements to delete");

        return;

    }

    for (i = 0; i <= rear; i++)

    {

        if (data == p\_queue[i])

        {

            for (; i < rear; i++)

            {

                p\_queue[i] = p\_queue[i + 1];

            }

        p\_queue[i] = -99;

        rear--;

        if (rear == -1)

            front = -1;

        return;

        }

    }

    printf("\n%d not found in queue to delete", data);

}

void display\_priorityqueue()

{

    if ((front == -1) && (rear == -1))

    {

        printf("\nQueue is empty");

        return;

    }

    for (; front <= rear; front++)

    {

        printf(" %d ", p\_queue[front]);

    }

    front = 0;

}

Output:-

1. Insert

2. Delete

3. Display

4. Quit

Enter your choice : 1

Enter value to be inserted : 2

Enter your choice : 1

Enter value to be inserted : 3

Enter your choice : 1

Enter value to be inserted : 4

Enter your choice : 1

Enter value to be inserted : 5

Enter your choice : 2

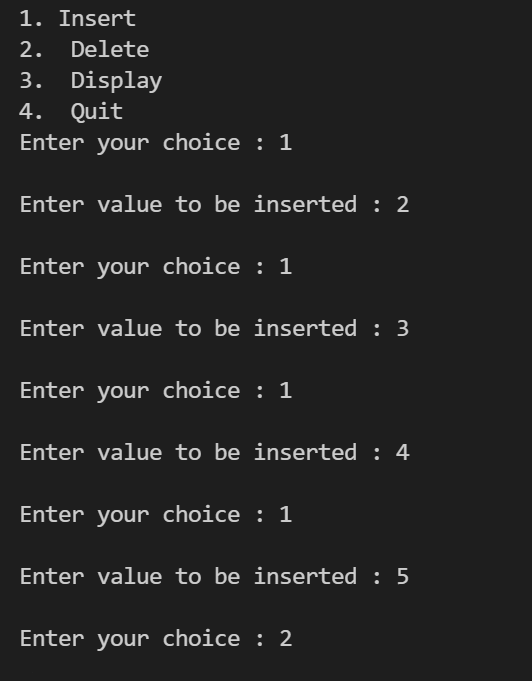
Enter value to delete : 2

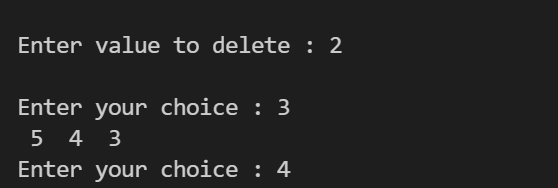
Enter your choice : 3

5 4 3

Enter your choice : 4

Screenshot:-





Q4. • WAP to implement Input-restricted De-Queue using array. Perform Insert°, delete°. and traverse 0 operation.

# include<stdio.h>

# define MAX 5

int deque\_arr[MAX];

int left = -1;

int right = -1;

void insert\_right()

{

    int added\_item;

    if((left == 0 && right == MAX-1) || (left == right+1))

    {   printf("Queue Overflow\n");

        return;}

    if (left == -1)  // initially empty

    {   left = 0;

        right = 0;}

    else

    if(right == MAX-1)  // last position is right

        right = 0;

    else

        right = right+1;

    printf("Input the element for adding in queue : ");

    scanf("%d", &added\_item);

    deque\_arr[right] = added\_item ;

}

void delete\_left()

{   if (left == -1)

    {   printf("Queue Underflow\n");

        return ;    }

    printf("Element deleted from queue is : %d\n",deque\_arr[left]);

    if(left == right) // agar ek hai tho

    {   left = -1;

        right=-1;    }

    else

        if(left == MAX-1)

            left = 0;

        else

            left = left+1;

}

void delete\_right()

{if (left == -1)

    {printf("Queue Underflow\n");

        return ;     }

    printf("Element deleted from queue is : %d\n",deque\_arr[right]);

    if(left == right) //agar sirf ek hai tho

    {   left = -1;

        right=-1;    }

    else

        if(right == 0)

            right=MAX-1;

        else

            right=right-1;  }

void display\_queue()

{   int front\_pos = left,rear\_pos = right;

    if(left == -1)

    {   printf("Queue is empty\n");

        return;  }

    printf("Queue elements :\n");

    if( front\_pos <= rear\_pos )

    {   while(front\_pos <= rear\_pos)

        {   printf("%d ",deque\_arr[front\_pos]);

            front\_pos++;    }   }

    else

    {   while(front\_pos <= MAX-1)

        {   printf("%d ",deque\_arr[front\_pos]);

            front\_pos++;    }

        front\_pos = 0;

        while(front\_pos <= rear\_pos)

        {   printf("%d ",deque\_arr[front\_pos]);

            front\_pos++;

        }

    }

    printf("\n");

}

void input\_que()

{   int choice;

    do

    {   printf("1.Insert at right\n");

        printf("2.Delete from left\n");

        printf("3.Delete from right\n");

        printf("4.Display\n");

        printf("5.Quit\n");

        printf("Enter your choice : ");

        scanf("%d",&choice);

        switch(choice)

        {   case 1:

            insert\_right();

            break;

         case 2:

            delete\_left();

            break;

         case 3:

            delete\_right();

            break;

         case 4:

            display\_queue();

            break;

         case 5:

            break;

         default:

            printf("Wrong choice\n");

        }

    }while(choice!=5);

}

main()

{   int choice;

    printf("/\*\*\*\*\*\*\*\*\*\*\*-----Input restricted dequeue------\*\*\*\*\*\*\*\*\*\*\*\*\n");

        input\_que();

    }

Output:-

/\*\*\*\*\*\*\*\*\*\*\*-----Input restricted dequeue------\*\*\*\*\*\*\*\*\*\*\*\*

1.Insert at right

2.Delete from left

3.Delete from right

4.Display

5.Quit

Enter your choice : 1

Input the element for adding in queue : 2

1.Insert at right

2.Delete from left

3.Delete from right

4.Display

5.Quit

Enter your choice : 1

Input the element for adding in queue : 3

1.Insert at right

2.Delete from left

3.Delete from right

4.Display

5.Quit

Enter your choice : 1

Input the element for adding in queue : 4

1.Insert at right

2.Delete from left

3.Delete from right

4.Display

5.Quit

Enter your choice : 1

Input the element for adding in queue : 5

1.Insert at right

2.Delete from left

3.Delete from right

4.Display

5.Quit

Enter your choice : 2

Element deleted from queue is : 2

1.Insert at right

2.Delete from left

3.Delete from right

4.Display

5.Quit

Enter your choice : 3

Element deleted from queue is : 5

1.Insert at right

2.Delete from left

3.Delete from right

4.Display

5.Quit

Enter your choice : 4

Queue elements :

3 4

1.Insert at right

2.Delete from left

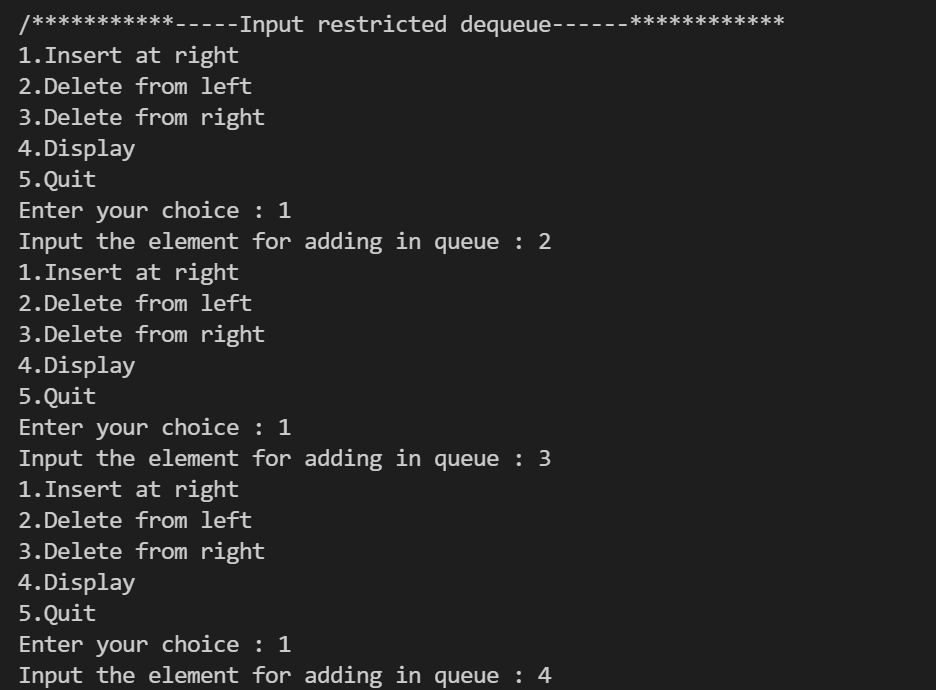
3.Delete from right

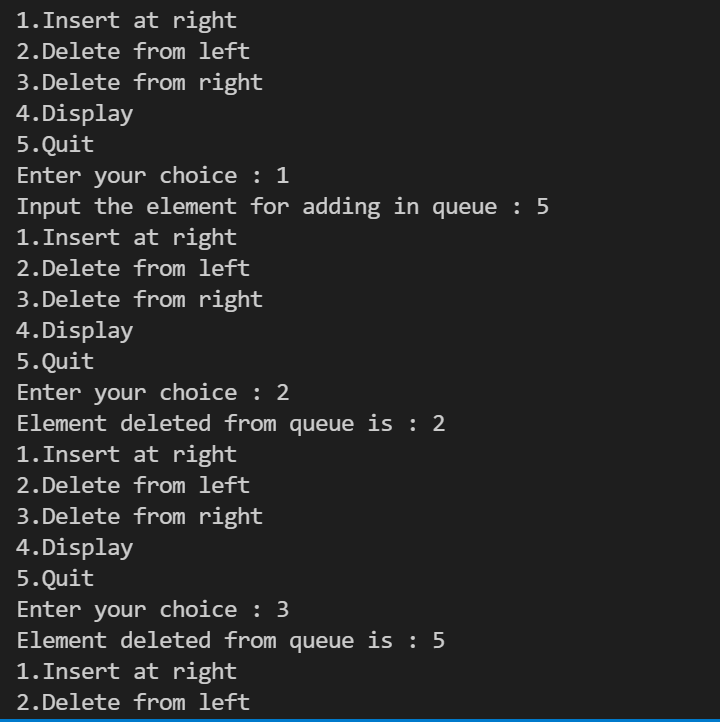
4.Display

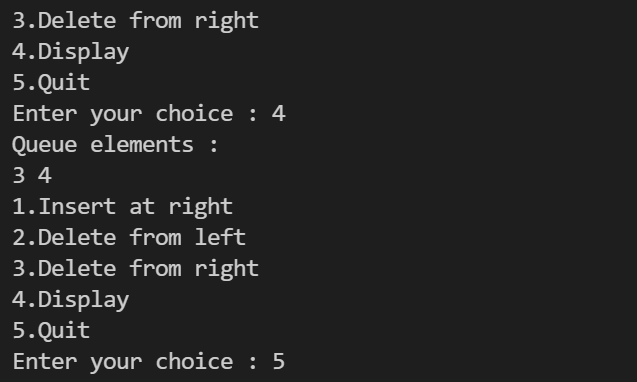
5.Quit

Enter your choice : 5

Screenshot:-



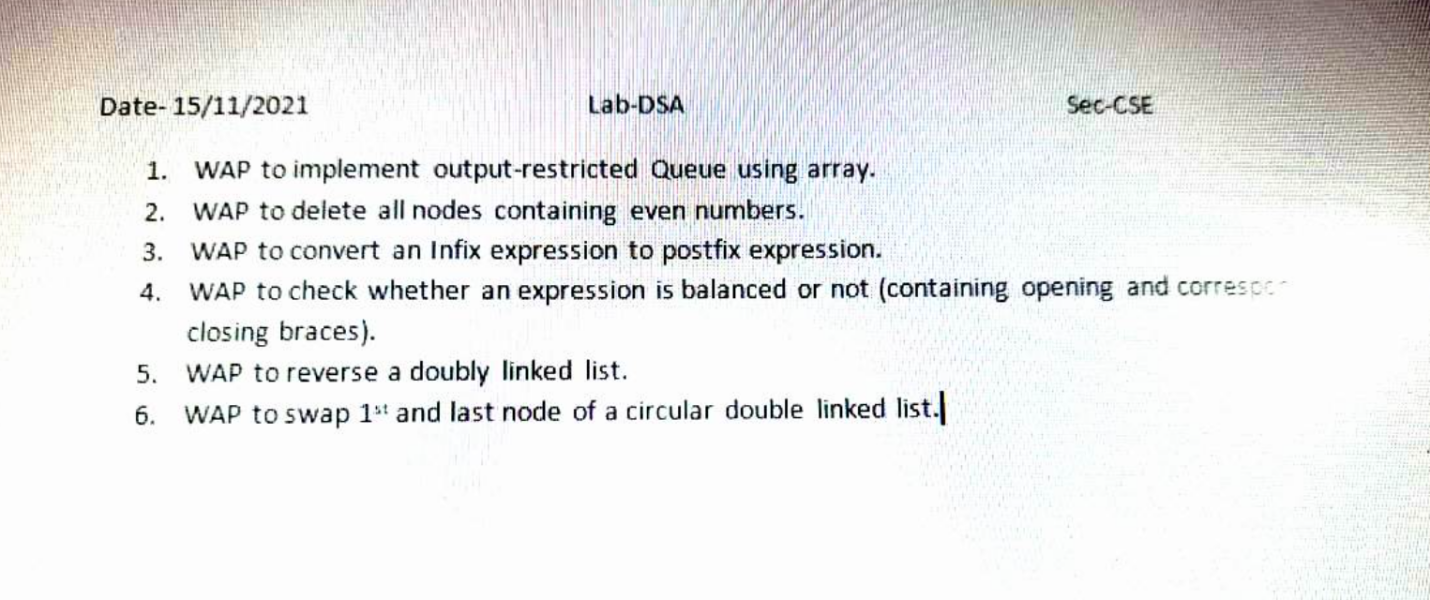




Lab 11

Date:- 13/11/2021

Q.



Q1. WAP to implement output-restricted Queue using array

Code:-

# include<stdio.h>

# define MAX 10

int deque\_arr[MAX];

int left = -1;

int right = -1;

void enqueue\_right()

{

    int insert\_item;

    if((left == 0 && right == MAX-1) || (left == right+1))

    {   printf("Queue Overflow\n");

        return;}

    if (left == -1)  // agar empty hai tho

    {   left = 0;

        right = 0;}

    else

    if(right == MAX-1)  // last postion right

        right = 0;

    else

        right = right+1;

    printf("Input the element for inserting in the queue : ");

    scanf("%d", &insert\_item);

    deque\_arr[right] = insert\_item ;

}

void enqueue\_left()

{   int added\_item;

    if((left == 0 && right == MAX-1) || (left == right+1))

    {   printf("Queue Overflow \n");

        return;  }

    if (left == -1)  //agar empty hua tho

    {   left = 0;

        right = 0;   }

    else

    if(left== 0)

        left=MAX-1;  //last position left

    else

        left=left-1;

    printf("Input the element for Inserting in the queue : ");

    scanf("%d", &added\_item);

    deque\_arr[left] = added\_item ;   }

void dequeue\_left()

{   if (left == -1)

    {   printf("Queue Underflow\n");

        return ;    }

    printf("The Element which you want to deleted from queue = %d\n",deque\_arr[left]);

    if(left == right) // only one ke liye

    {   left = -1;

        right=-1;    }

    else

        if(left == MAX-1)

            left = 0;

        else

            left = left+1;

}

void display\_queue()

{   int front\_pos = left,rear\_pos = right;

    if(left == -1)

    {   printf("Queue is empty\n");

        return;  }

    printf("The Queue elements are =\n");

    if( front\_pos <= rear\_pos )

    {   while(front\_pos <= rear\_pos)

        {   printf("%d ",deque\_arr[front\_pos]);

            front\_pos++;    }   }

    else

    {   while(front\_pos <= MAX-1)

        {   printf("%d ",deque\_arr[front\_pos]);

            front\_pos++;    }

        front\_pos = 0;

        while(front\_pos <= rear\_pos)

        {   printf("%d ",deque\_arr[front\_pos]);

            front\_pos++;

        }

    }

    printf("\n");

}

void output\_que()

{   int choice;

    do

    {   printf("1.Insert at right\n");

        printf("2.Insert at left\n");

        printf("3.Delete from left\n");

        printf("4.Display\n");

        printf("5.Quit\n");

        printf("Enter your choice : ");

        scanf("%d",&choice);

        switch(choice)

        {

         case 1:

            enqueue\_right();

            break;

         case 2:

            enqueue\_left();

            break;

         case 3:

            dequeue\_left();

            break;

         case 4:

            display\_queue();

            break;

         case 5:

         printf("\n\*\*\*\*\*\*\*\*You have quit the program succesfully\*\*\*\*\*\*\*\* ");

            break;

         default:

            printf("Wrong choice\n");

        }

    }while(choice!=5);

}

main()

{   int choice;

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*---------Output restricted dequeue----------\*\*\*\*\*\*\*\*\*\*\*\*\n");

    output\_que();

    }

Output:-

\*\*\*\*\*\*\*\*\*\*\*\*---------Output restricted dequeue----------\*\*\*\*\*\*\*\*\*\*\*\*

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 1

Input the element for inserting in the queue : 2

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 1

Input the element for inserting in the queue : 3

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 1

Input the element for inserting in the queue : 4

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 2

Input the element for Inserting in the queue : 5

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 2

Input the element for Inserting in the queue : 6

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 3

The Element which you want to deleted from queue = 6

1.Insert at right

2.Insert at left

3.Delete from left

4.Display

5.Quit

Enter your choice : 4

The Queue elements are =

5 2 3 4

1.Insert at right

2.Insert at left

3.Delete from left

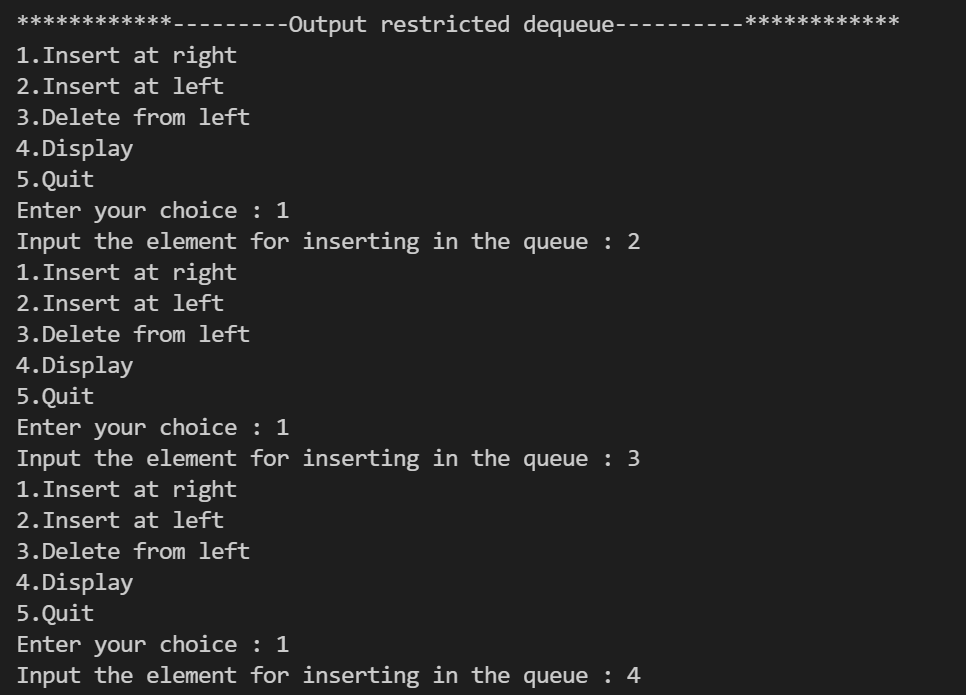
4.Display

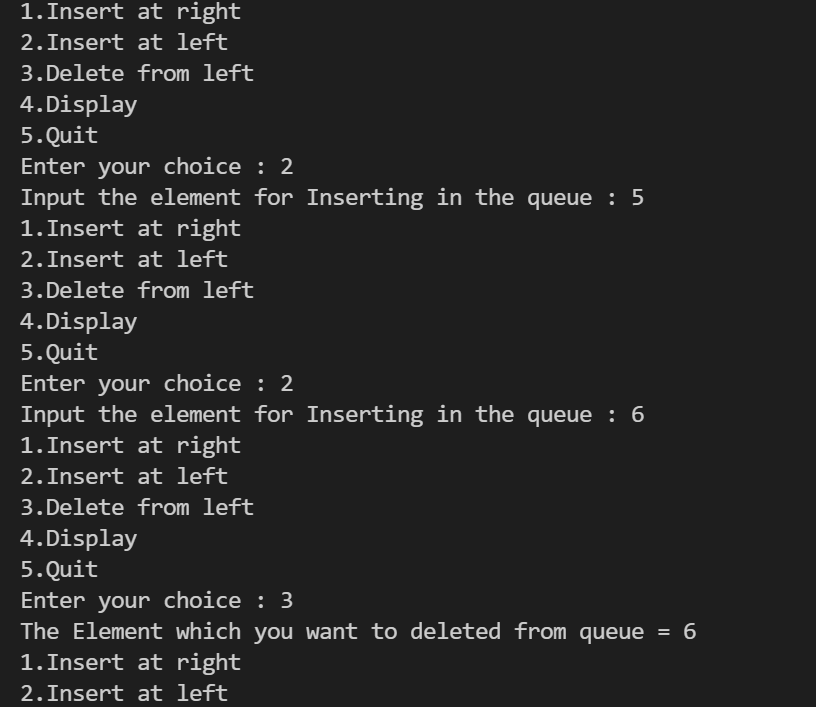
5.Quit

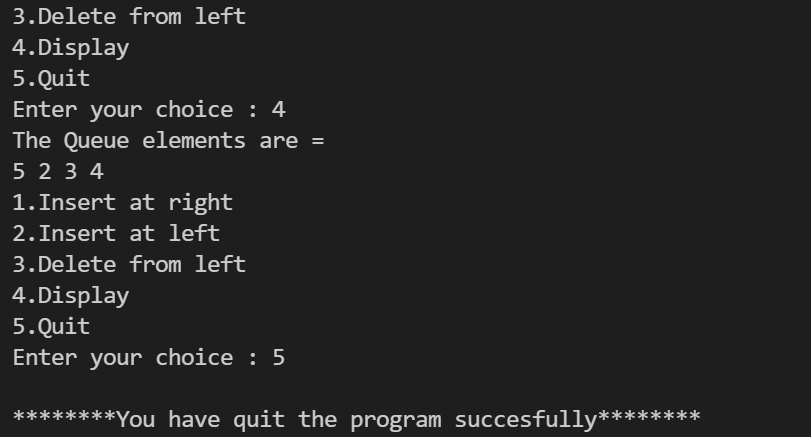
Enter your choice : 5

\*\*\*\*\*\*\*\*You have quit the program succesfully\*\*\*\*\*\*\*\*

Screenshot:-







Q2. WAP to delete all nodes containing even numbers.

#include <stdio.h>

#include <stdlib.h>

struct Node {

  int data;

  struct Node\* next;

};

void push\_back(struct Node\*\* head, int new) {

  struct Node \*newNode, \*temp;

  newNode = (struct Node\*)malloc(sizeof(struct Node));

  newNode->data = new;

  newNode->next = NULL;

  if(\*head == NULL)

   {

    \*head = newNode;

  }

  else

  {

    temp = \*head;

    while(temp->next != NULL) {

      temp = temp->next;

    }

    temp->next = newNode;

  }

}

void deleteEvenNodes(struct Node\*\* head) {

  if(\*head != NULL) {

    struct Node\* oddNode = \*head;

    struct Node\* evenNode = (\*head)->next;

    while(oddNode != NULL && evenNode != NULL) {

      oddNode->next = evenNode->next;

      free(evenNode);

      oddNode = oddNode->next;

      if(oddNode != NULL)

        evenNode = oddNode->next;

    }

  }

}

void display(struct Node\* head\_ref) {

  struct Node\* temp = head\_ref;

  if(head\_ref != NULL) {

    printf("Your node list contains =  ");

    while (temp != NULL) {

      printf("%i ",temp->data);

      temp = temp->next;

    }

    printf("\n");

  } else {

    printf("The list is empty.\n");

  }

}

int main() {

  struct Node\* a = NULL;

  push\_back(&a, 95);//96

  push\_back(&a, 96);//97

  push\_back(&a, 97);//98

  push\_back(&a, 98);//99

  push\_back(&a, 99);//100

  display(a);

  deleteEvenNodes(&a);

  printf("Even nodes deleted Succesfull.\n");

  display(a);

  return 0;

}

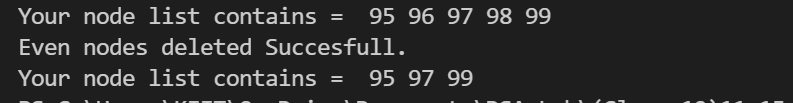
Output:-

Your node list contains = 95 96 97 98 99

Even nodes deleted Successful.

Your node list contains = 95 97 99

ScreenShot:-



Q3.WAP to convert an Infix expression to postfix expression.

#include <stdio.h>

char stack[10];

int top=-1;

push(char elem)

{                       // PUSH operation

    stack[++top]=elem;

}

char pop()

{                      // POP operation

    return(stack[top--]);

}

int precedence(char symbol)

{

    if(symbol == '^')

    {

        return(3);

    }

    else if(symbol == '\*' || symbol == '/')

    {

        return(2);

    }

    else if(symbol == '+' || symbol == '-')

    {

        return(1);

    }

    else

    {

        return(0);

    }

}

void main()

{

    char infinix[50],postfix[50],ch,ele;

    int i=0,k=0;

    printf("Enter a Infix Expression = ");

    scanf("%s",infinix);

    push('#');

    while( (ch=infinix[i++]) != '\0')

    {

        if( ch == '(') push(ch);

        else

            if(isalnum(ch)) postfix[k++]=ch;

            else

                if( ch == ')')

                {

                    while( stack[top] != '(')

                        postfix[k++]=pop();

                    ele=pop(); //removal

                }

                else

                {

                    while( precedence(stack[top]) >= precedence(ch) )

                        postfix[k++]=pop();

                    push(ch);

                }

    }

    while( stack[top] != '#')

        postfix[k++]=pop();

    postfix[k]='\0';

    printf("\nPostfix Expression =  %s\n",postfix);

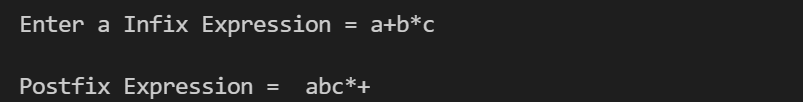
}

Ouput:-

Enter a Infix Expression = a+b\*c

Postfix Expression = abc\*+

Screenshot:-



Q4. WAP to check whether an expression is balanced or not (containing opening and corresponding - closing braces).

#include<stdio.h>

int main()

{

    char expression[10];

    int x=0, i=0;

    printf("\nEnter an expression  = ");

    scanf("%s", expression);

 while(expression[i]!= '\0')

    {

     if(expression[i]=='(')

        {

            x++;

        }

     else if(expression[i]==')')

        {

            x--;

            if(x<0)

            break;

        }

    i++;       // incrementing of i

    }

    if(x==0)

    {

        printf(" The Expression is balanced");

    }

    else

    {

        printf(" The Expression is unbalanced");

    }

    return 0;

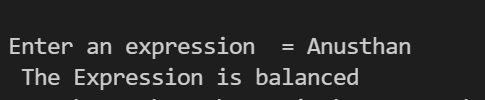
}

Output:-

Enter an expression = Anusthan

The Expression is balanced

Screenshot:-



Q5. WAP to reverse a doubly linked list.

Code:-

#include <stdio.h>

struct node{

    int data;

    struct node \*previous;

    struct node \*next;

};

struct node \*head, \*tail = NULL;

void addNode(int data) {

    //creating link list

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

    newNode->data = data;

    if(head == NULL) {

        head = tail = newNode;

        head->previous = NULL;

        tail->next = NULL;

    }

    else {

        tail->next = newNode;

        newNode->previous = tail;

        tail = newNode;

        tail->next = NULL;

    }

}

void reverse() {

    struct node \*current = head, \*temp = NULL;

    while(current != NULL) {

        temp = current->next;

        current->next = current->previous;

        current->previous = temp;

        current = current->previous;

    }

    temp = head;

    head = tail;

    tail = temp;

}

void display() {

    struct node \*current = head;

    if(head == NULL) {

        printf("List is empty\n");

        return;

    }

    while(current != NULL) {

        printf("%d ", current->data);

        current = current->next;

    }

}

int main()

{

    addNode(98);

    addNode(55);

    addNode(7);

    addNode(74);

    addNode(4);

    printf("Original List: \n");

    display();

    reverse();

    printf("\nReversed List: \n");

    display();

    return 0;

}

Output:-

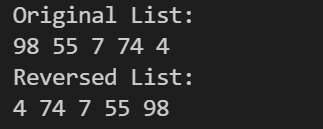
Original List:

98 55 7 74 4

Reversed List:

4 74 7 55 98

Screenshot:-



Q6.WAP to swap 1 and last node of a circular double linkrd list :

Code:-

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

void create(struct node \*\*head,int elements)

{

    struct node \*current,\*ptr;

    current=malloc(sizeof(struct node));

    printf("\nEnter the %d element : ",elements);

    scanf("%d",&current->data);

    current->next=NULL;

    if(\*head==NULL)

    {

        \*head=current;

        current->next=\*head;

        ptr=current;

    }

    else

    {

        ptr->next=current;

        current->next=\*head;

        ptr=current;

    }

}

void display(struct node \*h)

{

    struct node \*cur;

    cur=h;

    if(cur==NULL)

    {

      printf("\nThe of No nodes present");

      return;

    }

    printf("\n\nThe node are : ");

    while(cur->next!=h)

    {

        printf("%d ",cur->data);

        cur=cur->next;

    }

    printf("%d ",cur->data);

}

void swap\_last(struct node \*\*head)

{

    struct node \*cur,\*frwd;

    cur=\*head;

    if(cur==NULL || cur->next==\*head)

        return;

    while(cur->next->next!=\*head)

    {

        cur=cur->next;

    }

    frwd=cur->next;

    frwd->next=(\*head)->next;

    cur->next=\*head;

    (\*head)->next=frwd;

    \*head=frwd;

}

void main()

{

    struct node \*head=NULL;

    int n,i;

    printf("Enter the number of elements you want : ");

    scanf("%d",&n);

    for(i=0;i<n;i++)

    {

       create(&head,i+1);

    }

    printf("\nInitially Before Swapping the Nodes \n ");

    display(head);

    swap\_last(&head);

    printf("\nFirst and last Node swipped Successfully : ");

    display(head);

}

Output:-

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node \*next;

};

void create(struct node \*\*head,int elements)

{

    struct node \*current,\*ptr;

    current=malloc(sizeof(struct node));

    printf("\nEnter the %d element : ",elements);

    scanf("%d",&current->data);

    current->next=NULL;

    if(\*head==NULL)

    {

        \*head=current;

        current->next=\*head;

        ptr=current;

    }

    else

    {

        ptr->next=current;

        current->next=\*head;

        ptr=current;

    }

}

void display(struct node \*h)

{

    struct node \*cur;

    cur=h;

    if(cur==NULL)

    {

      printf("\nThe of No nodes present");

      return;

    }

    printf("\n\nThe node are : ");

    while(cur->next!=h)

    {

        printf("%d ",cur->data);

        cur=cur->next;

    }

    printf("%d ",cur->data);

}

void swap\_last(struct node \*\*head)

{

    struct node \*cur,\*frwd;

    cur=\*head;

    if(cur==NULL || cur->next==\*head)

        return;

    while(cur->next->next!=\*head)

    {

        cur=cur->next;

    }

    frwd=cur->next;

    frwd->next=(\*head)->next;

    cur->next=\*head;

    (\*head)->next=frwd;

    \*head=frwd;

}

void main()

{

    struct node \*head=NULL;

    int n,i;

    printf("Enter the number of elements you want : ");

    scanf("%d",&n);

    for(i=0;i<n;i++)

    {

       create(&head,i+1);

    }

    printf("\nInitially Before Swapping the Nodes \n ");

    display(head);

    swap\_last(&head);

    printf("\nFirst and last Node swipped Successfully : ");

    display(head);

}

Output:-

Enter the number of elements you want : 4

Enter the 1 element : 2

Enter the 2 element : 5

Enter the 3 element : 3

Enter the 4 element : 7

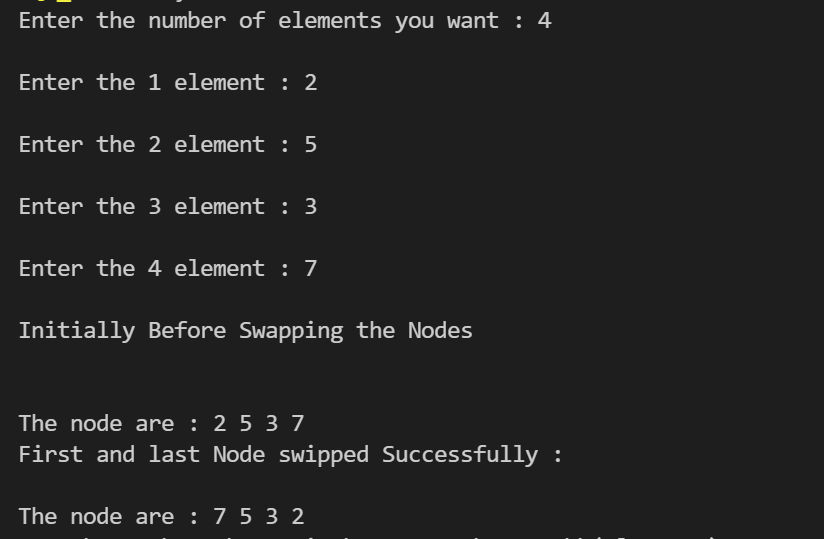
Initially Before Swapping the Nodes

The node are : 2 5 3 7

First and last Node swipped Successfully :

The node are : 7 5 3 2

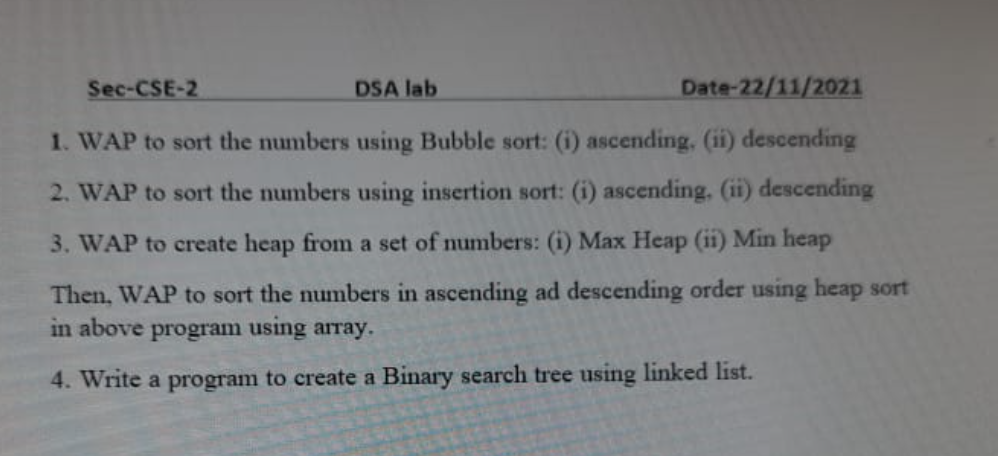
Screenshot:-



Lab 12

Date:- 22/11/2021

Q



Q1.WAP to sort the numbers using Bubble sort: (i) ascending. I descending

Code:-

#include <stdio.h>

void bubble\_sort\_ac(int [], int);

void bubble\_sort\_de(int [], int n);

int main()

{

  int array[10], n, i;

  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*Bubble Sort\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

  printf("\n\nEnter number of elements you want = \n");

  scanf("%ld", &n);

  printf("Enter %ld integers\n", n);

  for (i = 0; i < n; i++)

    scanf("%ld", &array[i]);

  bubble\_sort\_ac(array, n);

  printf(" Bubble Sorted list in ascending order:\n");

  for (i = 0; i < n; i++)

     printf("%ld\n", array[i]);

bubble\_sort\_de(array, n);

printf(" Bubble Sorted list in decending order:\n");

  for (i = 0; i < n; i++)

     printf("%ld\n", array[i]);

     printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*Bubble Sort Done succesfully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

  return 0;

}

void bubble\_sort\_ac(int list[], int n)

{

  int i, j, swap;

  for (i = 0 ; i < n - 1; i++) {

    for (j = 0 ; j < n - i - 1; j++) {

      if (list[j] > list[j+1]) {

        swap         = list[j];

        list[j]   = list[j+1];

        list[j+1] = swap;

      }

    }

  }

}

void bubble\_sort\_de( int list[], int n)

{

  int i, j, swap;

  for (i = 0 ; i < n - 1; i++) {

    for (j = 0 ; j < n - i - 1; j++) {

      if (list[j] < list[j+1]) {

        swap         = list[j];

        list[j]   = list[j+1];

        list[j+1] = swap;

      }

    }

  }

}

Output:-

\*\*\*\*\*\*\*\*\*\*\*\*\*Bubble Sort\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter number of elements you want =

4

Enter 4 integers

2

6

9

4

Bubble Sorted list in ascending order:

2

4

6

9

Bubble Sorted list in decending order:

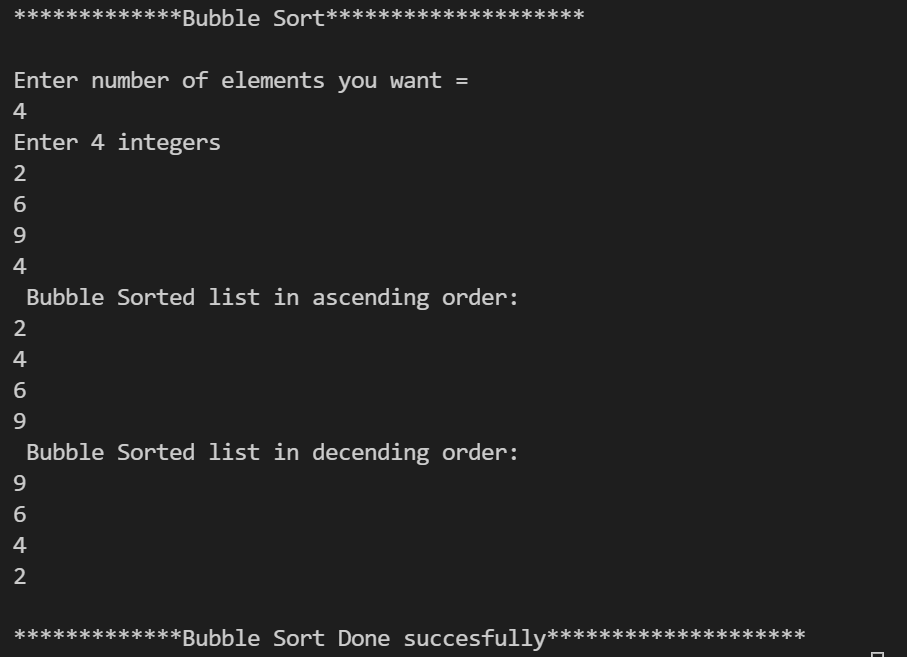
9

6

4

2

\*\*\*\*\*\*\*\*\*\*\*\*\*Bubble Sort Done succesfully\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Q2. WAP to sort the numbers using insertion sort: (i) ascending. (ii) descending

Code:-

#include<stdio.h>

void insert\_sort\_ac(int [], int );

void insert\_sort\_de(int [], int );

int main()

{

 int arr[10],i,n;

 printf("\n\t\*\*\*\*\*\*\*\*\*\*\*\*\*Insertion sort\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

 printf("Enter the number of elements you want = ");

 scanf("%d",&n);

 printf("Enter %d element : ",n);

 for(i=0; i<n; i++)

 {

    scanf("%d",&arr[i]);

 }

 insert\_sort\_ac(arr,n);

 printf("\nInsertion sorted list in ascending order : \n\n");

 for(i=0; i<n; i++)

    printf(" %d",arr[i]);

 insert\_sort\_de(arr,n);

 printf("\nInsertion sorted list in decending order :\n\n");

 for(i=0; i<n; i++)

    printf(" %d",arr[i]);

 return 0;

}

void insert\_sort\_ac(int arr[], int size)

{

 int i,j,tmp;

 for(i=0; i<size; i++)

 {

   for(j=i-1; j>=0; j--)

   {

    if(arr[j]>arr[j+1])

    {

      tmp=arr[j];

      arr[j]=arr[j+1];

      arr[j+1]=tmp;

    }

    else

      break;

   }

 }

}

void insert\_sort\_de(int arr[], int size)

{

 int i,j,tmp;

 for(i=0; i<size; i++)

 {

   for(j=i-1; j>=0; j--)

   {

    if(arr[j]<arr[j+1])

    {

      tmp=arr[j];

      arr[j]=arr[j+1];

      arr[j+1]=tmp;

    }

    else

      break;

   }

 }

}

Output:-

\*\*\*\*\*\*\*\*\*\*\*\*\*Insertion sort\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter the number of elements you want = 5

Enter 5 element : 2

7

4

1

9

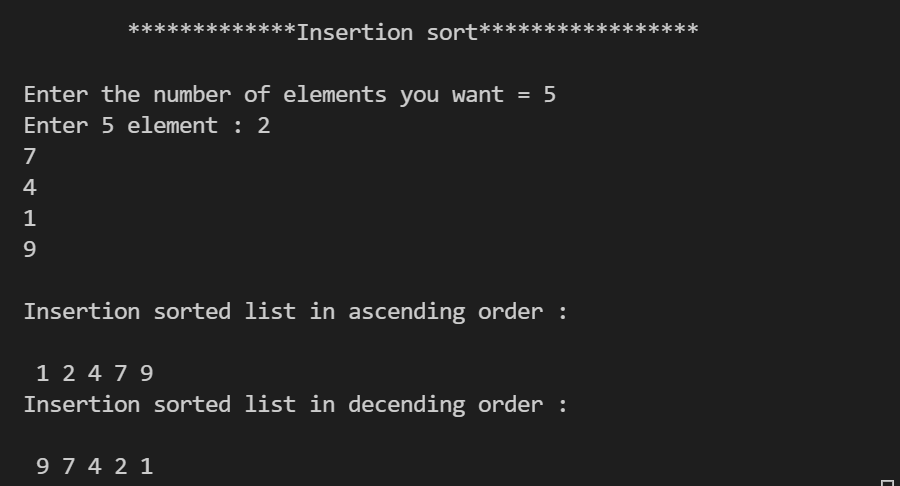
Insertion sorted list in ascending order :

1 2 4 7 9

Insertion sorted list in decending order :

9 7 4 2 1

Screenshot:-



Q3 WAP to create heap from a set of numbers: (i) Max Heap (ii) Min heap Then. WAP to sort the number in ascending ad descending order using heap sort in above program using array.

Code:-

#include <stdio.h>

void heapify(int a[], int n, int i)

{

    int largest = i;

    int left = 2 \* i + 1;

    int right = 2 \* i + 2;

    if (left < n && a[left] > a[largest])

        largest = left;

    if (right < n && a[right] > a[largest])

        largest = right;

    if (largest != i) {

        int temp = a[i];

        a[i] = a[largest];

        a[largest] = temp;

        heapify(a, n, largest);

    }

}

void heapSort(int a[], int n)

{

    for (int i = n / 2 - 1; i >= 0; i--)

        heapify(a, n, i);

    for (int i = n - 1; i >= 0; i--) {

        int temp = a[0];

        a[0] = a[i];

        a[i] = temp;

        heapify(a, i, 0);

    }

}

void ascending(int arr[], int n)

{

    for (int i = 0; i < n; ++i)

    {

        printf("%d", arr[i]);

        printf(" ");

    }

}

    void decending(int arr[], int n)

{

    for (int j = n-1; j>= 0; j--)

    {

        printf("%d", arr[j]);

        printf(" ");

    }

}

int main()

{

    int a[] = {10, 15, 16, 55, 99, 22, 11};

    int n = sizeof(a) / sizeof(a[0]);

    printf("Before sorting array elements : \n");

    ascending(a, n);

    heapSort(a, n);

    printf("\nAfter sorting array elements in ascending are : \n");

    ascending(a, n);

    printf("\n After sorting array elements in decending are :  \n");

    decending(a,n);

    return 0;

}

Output:-

Before sorting array elements :

10 15 16 55 99 22 11

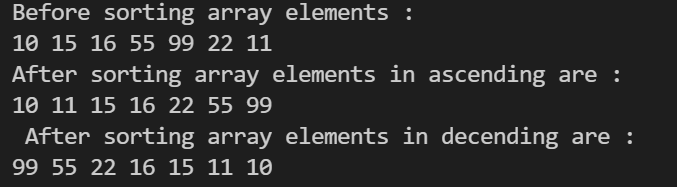
After sorting array elements in ascending are :

10 11 15 16 22 55 99

After sorting array elements in decending are :

99 55 22 16 15 11 10

Screenshot:-



Q4. Write a program to create a Binary search tree using linked list.

#include<stdio.h>

#include<malloc.h>

struct node

{

int data;

struct node \* left, \*right;

};

struct node\* create(struct node \*root,int n)

{

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

curr->data=n;

curr->left=curr->right=NULL;

if(root==NULL)

root=curr;

else

{

if(n<=root->data)

  root->left=create(root->left,n);

else

  root->right=create(root->right,n);

}

return root;

}

void inorder(struct node \*root)

 {

    if(root!=NULL)

  {

  inorder(root->left);

  printf("%d  ",root->data);

  inorder(root->right);

 }

 }

 void prorder(struct node \*root)

 {

  if(root!=NULL)

  {

  printf("%d  ",root->data);

  prorder(root->left);

  prorder(root->right);

  }

 }

void postorder(struct node \*root)

 {

    if(root!=NULL)

  {

  postorder(root->left);

  postorder(root->right);

  printf("%d  ",root->data);

 }

 }

 int main()

 {

        printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*BST(Binary Search Tree)\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

  struct node \*root=NULL;

  int value,choice=1;

  while(choice)

  {

    printf("\nEnter the value of the node = ");

    scanf("%d",&value);

    root=create(root,value);

    printf("\nDo you want to insert more nodes If Yes the Press 1 or No then Press 0 = ");

    scanf("%d",&choice);

   }

   printf("\n Your In-Order Traversal  = ");

   inorder(root);

   printf("\nYour Pre-Order Traversal  = ");

   prorder(root);

   printf("\nYour Post-Order Traversal  = ");

   postorder(root);

 }

Output:-

\*\*\*\*\*\*\*\*\*\*\*\*BST(Binary Search Tree)\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter the value of the node = 5

Do you want to insert more nodes If Yes the Press 1 or No then Press 0 = 1

Enter the value of the node = 2

Do you want to insert more nodes If Yes the Press 1 or No then Press 0 = 1

Enter the value of the node = 7

Do you want to insert more nodes If Yes the Press 1 or No then Press 0 = 1

Enter the value of the node = 8

Do you want to insert more nodes If Yes the Press 1 or No then Press 0 = 1

Enter the value of the node = 9

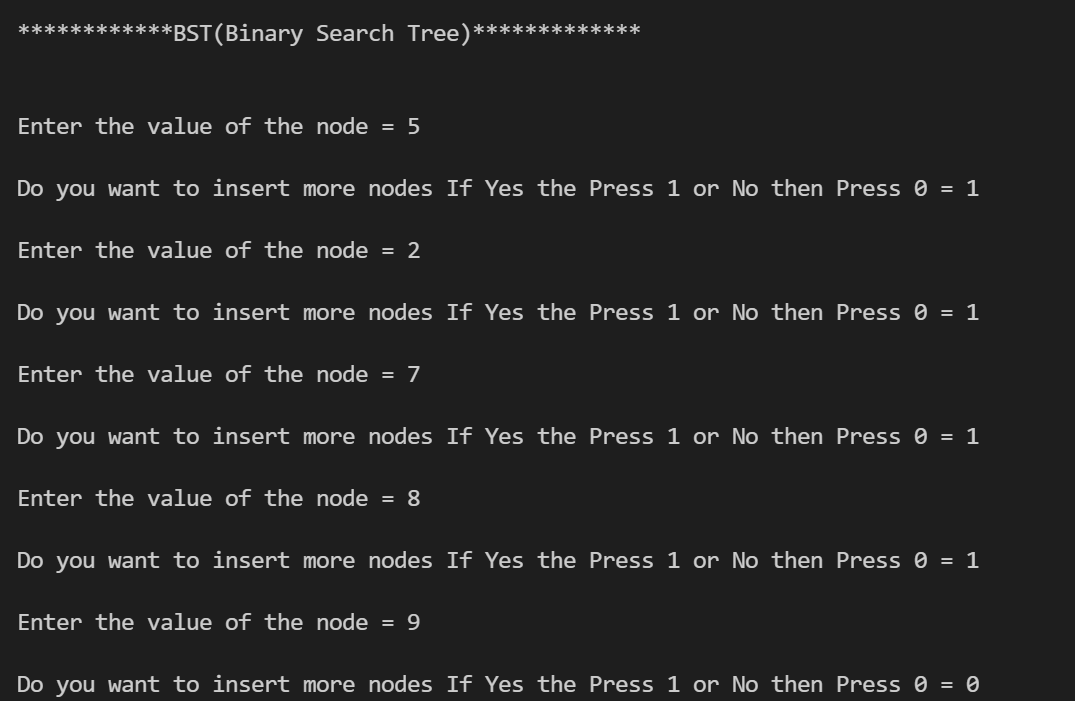
Do you want to insert more nodes If Yes the Press 1 or No then Press 0 = 0

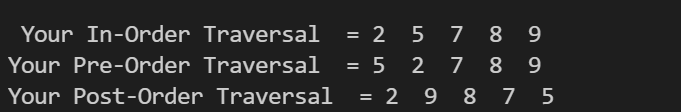
Your In-Order Traversal = 2 5 7 8 9

Your Pre-Order Traversal = 5 2 7 8 9

Your Post-Order Traversal = 2 9 8 7 5

Screenshot:-





By:- Anusthan Singh 20051337

DSA LAB FINAL SUBMITION