**DS Project File**

Name: Patel Hetvi.

En. No.: 210410107004

Class: SY-I

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Title | Page No. | Remarks |
| 1. | Program to illustrate call by value and call by reference. |  |  |
| 2. | Program to store data in matrix in irregular columns. |  |  |
| 3. | Program to convert infix expression to postfix without parentheses. |  |  |
| 4. | Program to convert infix expression into postfix expression with parentheses. |  |  |
| 5. | Program to implement Simple Queue. |  |  |
| 6. | Program to implement Circular Queue. |  |  |
| 7. | Program to implement Singly Linked List using menu. |  |  |
| 8. | Program to implement insertion and deletion operation on Ordered Linked List. |  |  |
| 9. | Program to implement Circular Linked List. |  |  |
| 10. | Program to implement Doubly Linked List. |  |  |
| 11. | Program to implement Bubble Sort. (Ascending order) |  |  |
| 12. | Program to implement Quick Sort. |  |  |
| 13. | Program to implement Merge Sort. |  |  |
| 14. | Program to implement Linear Search. |  |  |
| 15. | Program to implement Binary Search. |  |  |

**1. Program to illustrate call by value and call by reference.**

Code:

# include<stdio.h>

void fun(int,int);

void fun1(int\*,int\*);

int main(){

printf("\nEn.No.:210410107004\nName:Patel Hetvi");

int A=10,B=20;

printf("\nCALL BY VALUE:");

printf("\n Values before calling %d, %d",A,B);

fun(A,B);

printf("\n Values after calling %d, %d",A,B);

printf("\n\nCALL BY REFERENCE:");

printf("\n Values before calling %d, %d",A,B);

fun1(&A,&B);

printf("\n Values after calling %d, %d",A,B);

return 0;

}

void fun(int X,int Y){

X++;

Y+=2;

}

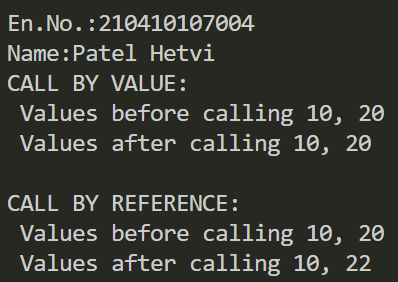
void fun1(int \*X,int \*Y){

\*X++;

\*Y+=2;

}

Output:



**2.Program to store data in matrix in irregular columns.**

Code:

#include<stdio.h>

#include<malloc.h>

int main()

{

printf("En.No.:210410107004 \nName:Patel Hetvi\n");

int \*\*arr;

int r,\*c;//c is array.

printf("How many rows you want?");

scanf("%d",&r);

//allocating row memory.

arr=(int\*\*)malloc(r\*sizeof(int\*));

c=(int\*)malloc(r\*sizeof(int));

//allocating column memory for each row.

for (int ri=0;ri<r;ri++){

printf("How many columns you want in row %d:",ri);

scanf("%d",&c[ri]);

arr[ri]=(int\*)malloc(c[ri]\*sizeof(int));

}//entering the numbers in array.

for (int i=0;i<r;i++){

for(int j=0;j<c[i];j++){

printf("Enter the arr[%d][%d]:",i,j);

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

}

}

printf("Array:\n");

for (int i=0;i<r;i++){

for(int j=0;j<c[i];j++){

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

}

printf("\n");

}

for (int ri=0;ri<r;ri++){

free(arr[ri]);

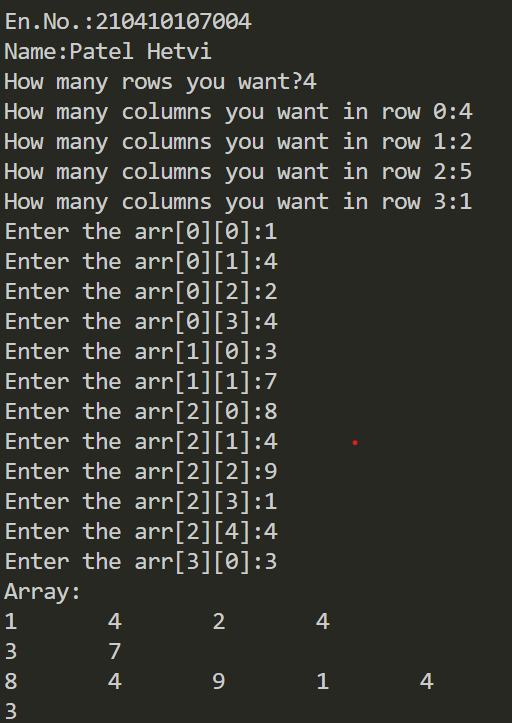
}

free(arr);

return 0;

}

Output:



**3.Program to convert infix expression to postfix without parentheses.**

Code:

Top of Form

#include<stdio.h>

#define MAX 10

char stk[MAX];

int top=-1;

push(char c){//check overflow

if (top>9){

printf("Overflow");

}// append in stack

else{

stk[++top]=c;}

}

char pop(){//check underflow

return stk[top--];

}

int pri(char x){

switch(x){

case '$':

return 0;

case '+':

return 1;

case '-':

return 1;

case '\*':

case '/':

return 2;

default:

return 3;

}

}

int main(){

printf("En.No.:210410107004 \nName:Patel Hetvi\n");

char input[15];

char output[15];

int curr=0,oind=0;

push('$');

scanf("%s",input);

while (input[curr]!='\0'){

while (pri(input[curr])<=pri(stk[top])){

output[oind]=pop();

oind++;

}

push(input[curr]);

curr++;

}

while (stk[top]!='$'){

output[oind]=pop();

oind++;

}

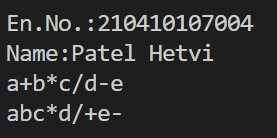
output[oind]='\0';

printf("%s \n",output);

return 0;

}

Output:



**4. Program to convert infix expression into postfix expression with parentheses.**

Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct stac{

char st[50];

int top;

};

void push(struct stac \*s1,char value){

s1->st[++(s1->top)]=value;

}

char pop(struct stac \*s1){

return s1->st[(s1->top)--];

}

int fprio(char ch){

switch(ch){

case '+':

case '-':

return 1;

case '\*':

case '/':

return 3;

case '^':

return 6;

case '(':

return 9;

case ')':

return 0;

default:

return 7;

}

}

int gprio(char ch){

switch(ch){

case '+':

case '-':

return 2;

case '\*':

case '/':

return 4;

case '^':

return 5;

case '(':

return 0;

default:

return 8;

}

}

struct stac myst;

void in2post(char input[],char output[]){

int indx=0,outindx=0;

push(&myst,'(');

while(input[indx]!='\0'){

while(fprio(input[indx])<gprio(myst.st[myst.top])){

output[outindx++]=pop(&myst);

}

if (fprio(input[indx])!=gprio(myst.st[myst.top])){

push(&myst,input[indx]);

}

else{

pop(&myst);

}

indx++;

}

output[outindx]='\0';

printf("%s\n",output);

}

int main()

{printf("enroll no.:210410107004\nName:Patel Hetvi \n");

myst.top=-1;

char input[]="(a+b^c^d)\*(e+f/g))";

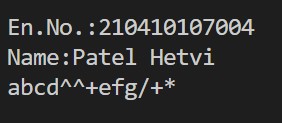
char output[20];

in2post(input,output);

return 0;

}

Output:



**5. Program to implement simple queue.**

Code:

#include<stdio.h>

#define M 5

int arr[M];

int f=-1;

int r=-1;

//insertion of element.

void qin(int a){

//check overflow

if(r==M){

printf("Overflow");

return;

}

r++;

arr[r]=a;

if (f==-1){

f=0;

}

}//deletion of element.

int qde(){

if (f==-1){

printf("Underflow \n");

return -1;

}

int temp;

temp=arr[f];

if (f==r){

f=r=-1;

}

else{

f++;

}

return temp;

}//Display the queue.

void disp(){

printf("Queue is:");

for(int i=f;i<=r;i++){printf("\n %d",arr[i]);}

}

int main(){

printf("En. No.:210410107004 \nName:Patel Hetvi \n");

qin(1);

qin(2);

printf(" %d\n",qde());

printf(" %d\n",qde());

printf(" %d\n",qde());

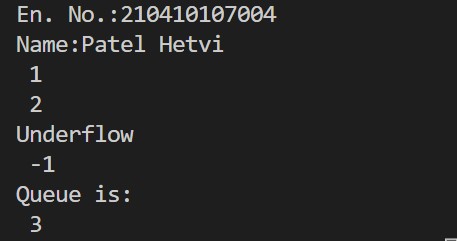
qin(3);

disp();

return 0;

}

Output:



**6. Program to implement circular queue.**

Code:

#include<stdio.h>

#define MAXSIZE 5

int arr[MAXSIZE];

int f=-1;

int r=-1;//insertion of data.

void cin(int a){

//overflow condition

if((r==MAXSIZE-1 && f==0)||(f==r+1)){

printf("Overflow!! \n");

return 0;

}

if (r==MAXSIZE-1){

r=0;

}

else{

r++;

}

arr[r]=a;

if (f==-1){

f=0;

}

}//deletion of data.

int cde(){

int temp;

if (f==-1){

printf("Underflow!!\n");

return -1;

}

temp=arr[f];

if (f==r){

f=r=0;

}

else{

if(f==MAXSIZE-1){f=0;}

else{f++;}

}

return temp;

}

int main(){

printf("Name: Hetvi Patel \n En. No. 210410107004 \n");

cin(1);

cin(2);

cin(3);

cin(4);

cin(5);

cin(6);

cin(7);

cin(8);

printf("%d\n",cde());

printf("%d\n",cde());

printf("%d\n",cde());

printf("%d\n",cde());

cin(9);

cin(10);

cin(11);

cin(12);

cin(13);

printf("%d\n",cde());

printf("%d\n",cde());

printf("%d\n",cde());

printf("%d\n",cde());

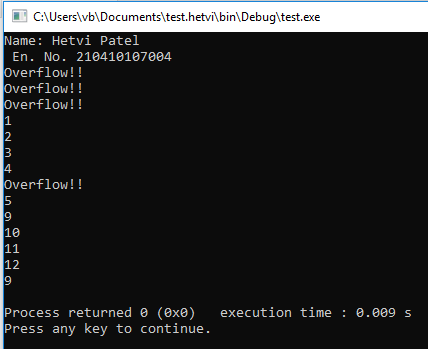
printf("%d\n",cde());

printf("%d\n",cde());

return 0;

}

Output:



**7. Program to implement singly linked list using menu.**

Code:

#include <stdio.h>

#include <stdlib.h>

struct ND{

int info;

struct ND\* next;

};//inserting an element in the beginning.

struct ND\* inbeg(struct ND\* first,int X){

//allocate node

struct ND\* newnd;

newnd=(struct ND\*)malloc(sizeof(struct ND));

if (newnd==NULL){

printf("Some Error!! \n");

}

//assign value

newnd->info=X;

newnd->next=NULL;

if (first==NULL){

first=newnd;

}

else{

newnd->next=first;

first=newnd;

}

};//deleting an element from the front.

struct ND\* delbeg(struct ND\* first){

if (first==NULL){

printf("UNDERFLOW\n");

return first;

}

struct ND\* temp=first;

first=first->next;

free(temp);

return first;

};//inserting an element in the end.

struct ND\* inend(struct ND\* first,int X){

//allocate node

struct ND\* newnd1,\*curr;

newnd1=(struct ND\*)malloc(sizeof(struct ND));

newnd1->info=X;

newnd1->next=NULL;

if (first==NULL){

return newnd1;

}

for(curr=first;curr->next!=NULL;curr=curr->next);

curr->next=newnd1;

//assign value

return first;

};//deleting an element from the end,

struct ND \*delast(struct ND \*first){

struct ND \*prev;

if(first==NULL){

printf("UNDERFLOW\n");

return first;

}

else if(first->next==NULL){

struct ND \*temp=first;

first=NULL;

free(temp);

return first;

}

else{

struct ND \*temp=first;

while(temp->next!=NULL){

prev=temp;

temp=temp->next;

}

prev->next=NULL;

free(temp);

return first;

}

};//deleting the element before specified position

struct ND\* delposbefore(struct ND \*first,int y){

int i;

int n=y;

struct ND \*temp=first;

struct ND \*prev;

if(temp==NULL){

printf("LIST IS EMPTY!! Kindly add elements..\n");

return first;

}

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

prev=temp;

temp=temp->next;

}

prev->next=temp->next;

free(temp);

return first;

}

struct ND\* delposafter(struct ND \*first,int y){

int i;

int n=y;

struct ND \*temp=first;

struct ND \*prev;

if(temp==NULL){

printf("LIST IS EMPTY!! Kindly add elements..\n");

return first;

}

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

prev=temp;

temp=temp->next;

}

prev->next=temp->next;

free(temp);

return first;

}

void disp(struct ND\* first){

if (first==NULL){

printf("List is empty!!");

}

else{

struct ND\*curr;

curr=first;

while(curr!=NULL){

printf("%d \n",curr->info);

curr=curr->next;

}

}

}

int main()

{

printf("En. No. 210410107004\nName:Hetvi Patel\n");

struct ND\* first=NULL;

printf("1.Insert a node at the front of the linked list.\n2.Insert a node at the end of the linked list.\n");

printf("3.Delete a first node of the linked list.\n4.Delete a node before specified position.\n");

printf("5.Delete a node after specified position.\n6.Display the list.\n");

int choice=0,value,position;

while(1){

printf("Enter your choice:");

scanf("%d",&choice);

if (choice==7){break;}

else{

switch (choice){

case 1:

printf("Enter the value you want to insert(beg):");

scanf("%d",&value);

first=inbeg(first,value);

break;

case 2:

printf("Enter the value you want to insert(last):");

scanf("%d",&value);

first=inend(first,value);

break;

case 3:

first=delbeg(first);

break;

case 4:

printf("Enter the position whose preceding element is to be deleted:");

scanf("%d",&position);

first=delposbefore(first,position);

break;

case 5:

printf("Enter the position whose succeeding element is to be deleted:");

scanf("%d",&position);

first=delposafter(first,position);

break;

case 6:

disp(first);

break;

default:

printf("Invalid choice!!\n");

break;

}

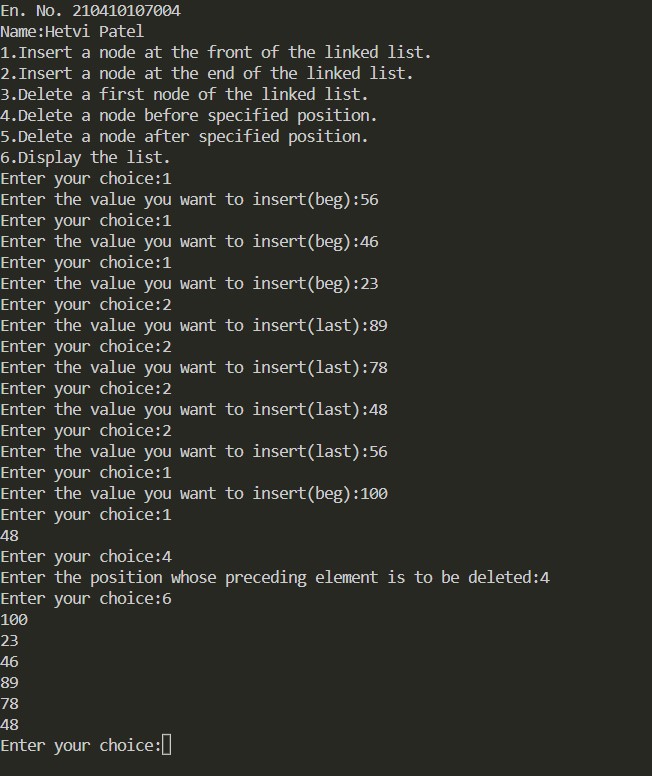
}

}

return 0;

}

Output:



**8. Program to implement insertion and deletion operation on ordered linked list.**

Code:

#include <stdio.h>

#include <stdlib.h>

struct node{

int info;

struct node\* next;

};//insertion of element in ordered list

struct node\* insertor(struct node\* first, int value){

struct node\* new\_node;

new\_node = (struct node\*)malloc(sizeof(struct node));

if (new\_node == NULL)

{

printf("Somethings wrong with malloc");

return;

}

new\_node->info=value;

new\_node->next=NULL;

if (first==NULL){

first=new\_node;

return first;

}

if (new\_node->info<= first->info){

new\_node->next=first;

first=new\_node;

return first;

}

struct node\* curr=first;

while (curr->next!=NULL && (new\_node->info>=curr->next->info)){

curr=curr->next;

}

new\_node->next=curr->next;

curr->next=new\_node;

return first;

};//display the list

void traverse(struct node\* first){

struct node\* curr;

if(first==NULL){

printf("The list is empty");

return;

}

for(curr=first;curr!=NULL;curr=curr->next){

printf("%d\n",curr->info);

}

}//deleting an element in list.

struct node\* deleteor(struct node\* first,int value){

if(first==NULL){

printf("The list is empty");

return;

}

struct node\* temp=first;

if (value==first->info){

first=first->next;

return first;

}

struct node\* curr=first;

struct node\* prev;

while(value!=curr->info){

prev=curr;

curr=curr->next;

}

prev->next=curr->next;

free(curr);

return first;

}

int main()

{

printf("Enrollment No.:210410107004\nName:Patel Hetvi\n");

struct node\* first = NULL;

first = insertor(first,1);

first = insertor(first,0);

first = insertor(first,2);

first = insertor(first,3);

first = insertor(first,4);

traverse(first);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

first=deleteor(first,0);

first=deleteor(first,3);

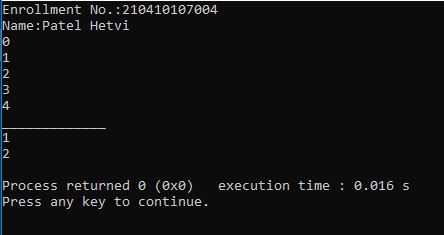
first=deleteor(first,4);

traverse(first);

return 0;

}

Output:



**9. Program to implement circular linked list.**

Code:

#include<stdio.h>

#include<stdlib.h>

struct ND{

int info;

struct ND \* next;

};//inserting an element in the front.

struct ND \*inbeg(struct ND \*first,int X){

struct ND \*newnode;

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

if(newnode==NULL){

printf("Allocation Error!!");

return first;

}

newnode->info=X;

newnode->next=NULL;

if(first==NULL){

newnode->next=newnode;

first=newnode;

}

else{

newnode->next=first;

struct ND \*curr=first;

while(curr->next!=first){

curr=curr->next;

}

curr->next=newnode;

first=newnode;

return first;

}

};//inserting an element in the end.

struct ND \*inlast(struct ND \*first,int X){

struct ND \*newnode;

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

if(newnode==NULL){

printf("Allocation Error!!");

return first;

}

newnode->info=X;

newnode->next=NULL;

if(first==NULL){

newnode->next=newnode;

first=newnode;

}

else{

struct ND \*curr=first;

while(curr->next!=first){

curr=curr->next;

}

curr->next=newnode;

newnode->next=first;

return first;

}

}//deleting an element from the front.

struct ND \*delbeg(struct ND \*first){

struct ND \*temp,\*curr,\*prev;

curr=first;

if(first==NULL){

printf("Empty List!!\n");

return first;

}//if there is only one element in list:

if(first->next==first){

temp=first;

first=NULL;

free(temp);

return first;

}

//if there are more than one nodes ..

else{

while(curr->next!=first){

curr=curr->next;

}

curr->next=first->next;

first=first->next;

free(temp);

return first;

}

}//deletion from last

struct ND \*delast(struct ND \*first){

struct ND \*temp,\*curr,\*prev;

curr=first;

if(first==NULL){

printf("Empty List!!\n");

return first;

}//if there is only one element in list:

if(first->next==first){

temp=first;

first=NULL;

free(temp);

return first;

}

//if there are more than one nodes ..

else{

while(curr->next!=first){

prev=curr;

curr=curr->next;

}

temp=curr;

prev->next=first;

free(temp);

return first;

}

}//displaying the list.

void traverse(struct ND \*first){

if(first==NULL){printf("Empty List!!\n");

}

else{

struct ND \*curr=first;

while(curr->next!=first){

printf("%d\n",curr->info);

curr=curr->next;

}

printf("%d\n",curr->info);

}

}

int main(){

printf("En.No.:210410107004\nName:Patel Hetvi\n");

struct ND \*first=NULL;

first=inbeg(first,1);

first=inbeg(first,2);

first=inbeg(first,3);

first=inbeg(first,4);

first=inbeg(first,5);

first=inlast(first,20);

first=inlast(first,10);

traverse(first);

printf("Deletion operation\n");

first=delbeg(first);

first=delbeg(first);

first=delbeg(first);

first=delast(first);

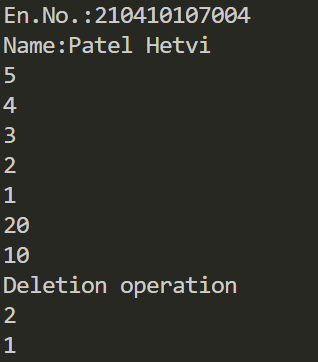
first=delast(first);

traverse(first);

return 0;

}

Output:



**10. Program to implement doubly linked list.**

Code:

#include<stdio.h>

#include<stdlib.h>

struct ND{

int info;

struct ND \*next,\*prev;

};

struct ND \*inbeg(struct ND \*first,int X){

struct ND \*nn;

nn=(struct ND \*)malloc(sizeof(struct ND));

if (nn==NULL){

printf("Allocation Error!!");

return first;

}

nn->info=X;

nn->prev=NULL;

nn->next=NULL;

if(first==NULL){

first=nn;

return first;

}

else{

nn->next=first;

first->prev=nn;

first=nn;

return first;

}

};

struct ND \*inend(struct ND \*first,int X){

struct ND \*nn,\*temp;

temp=first;

nn=(struct ND \*)malloc(sizeof(struct ND));

if (nn==NULL){

printf("Allocation Error!!");

return first;

}

nn->info=X;

nn->prev=NULL;

nn->next=NULL;

if(first==NULL){

first=nn;

return first;

}

while(temp->next!=NULL){

temp=temp->next;

}

nn->prev=temp;

temp->next=nn;

return first;

};

struct ND \*delbeg(struct ND \*first){

if(first==NULL){

printf("EMPTY LIST\n");

return first;

}

if(first->next==NULL){

struct ND \*temp=first;

first=NULL;

free(temp);

return first;

}

else{

struct ND \*temp=first;

first=first->next;

first->prev=NULL;

free(temp);

return first;

}

};

struct ND \*delast(struct ND \*first){

if(first==NULL){

printf("UNDERFLOW\n");

return first;

}

else if(first->next==NULL){

struct ND \*temp=first;

first=NULL;

free(temp);

return first;

}

else{

struct ND \*temp=first;

while(temp->next!=NULL){

temp=temp->next;

}

temp->prev->next=NULL;

free(temp);

return first;

}

};

void delposbefore(struct ND \*\*first,int y){

int i=0;

int n=y;

struct ND \*temp=\*first;

struct ND \*del;

if(temp==NULL){

printf("LIST IS EMPTY!! Kindly add elements..\n");

return; }

while(i!=n){

temp=temp->next;

if(temp==NULL){

printf("End of list!!!\n");

return;

}

i++;

}

del=temp;

(temp->prev->next)=temp->next;

(temp->next->prev)=temp->prev;

free(temp);

}

void traverse(struct ND \*first){

struct ND \*curr;

if(first==NULL){

printf("Empty list.\n");

return;

}

curr=first;

while(curr->next!=NULL){

printf("%d\n",curr->info);

curr=curr->next;

}

printf("%d\n",curr->info);}

int main(){

printf("\nEn.No.:210410107004\nName:Patel Hetvi\n");

struct ND \*first=NULL;

int choice=0,value,position;

printf("1.Insert a node at the front of list\n2.Insert a node at the end of list\n");

printf("3.Delete a last node\n4.Delete a node before a specified position.\n5.To display the list\n6.EXIT\n\n");

while(1){

printf("Enter your choice:");

scanf("%d",&choice);

if (choice==6){break;}

else{

switch (choice){

case 1:

printf("Enter the value you want to insert(beg):");

scanf("%d",&value);

first=inbeg(first,value);

break;

case 2:

printf("Enter the value you want to insert(last):");

scanf("%d",&value);

first=inend(first,value);

break;

case 3:

first=delbeg(first);

break;

case 4:

printf("Enter the position whose preceding element is to be deleted:");

scanf("%d",&position);

delposbefore(&first,position);

break;

case 5:

traverse(first);

break;

default:

printf("Invalid choice!!\n");

break;

}

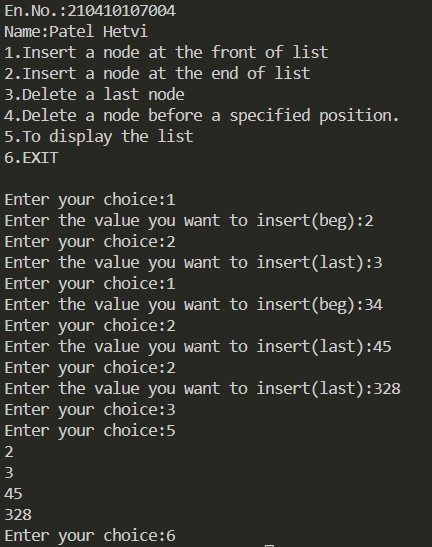
}

}

return 0;

}

Output:



**11 Program to implement bubble sort. (ascending order)**

Code:

#include<stdio.h>

void swap(int \*x,int \*y){

int temp;

temp=\*x;

\*x=\*y;

\*y=temp;

return;

}

void bs(int arr[],int n){

int flag=0;

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

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

if(arr[j]>arr[j+1]){

swap(&arr[j],&arr[j+1]);

flag=1;

}

}

if(flag==0){

printf("Array is already sorted!!\n");

return;

}

}

return;

}

void disp( int arr[],int size){

for(int m=0;m<size;m++){

printf("%d\_",arr[m]);

}

printf("\n");

return;

}

int main(){

printf("En.No.:210410107004\nName:Patel Hetvi\n");

int MS;

printf("Enter the number of elements to be sorted:");

scanf("%d",&MS);

int arr[MS];

for (int w=0;w<MS;w++){

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

}

printf("before:");

disp(arr,MS);

bs(arr,MS);

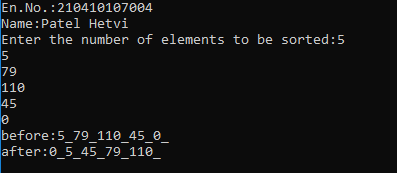
printf("after:");

disp(arr,MS);

return 0;

}

Output:



**12. Program to implement quick sort.**

Code:

#include<stdio.h>

void swap(int \*x,int \*y){

int temp=\*x;

\*x=\*y;

\*y=temp;

}//partition

int part(int arr[],int low,int high){

int pi=high;

int i=low,j;

for(j=low;j<high;j++){

if(arr[j]<=arr[pi]){

swap(&arr[i],&arr[j]);

i++;

}// the array with elements less than pivot before pivot and elements greater than pivot after pivot.

}

swap(&arr[i],&arr[pi]);

return i;

}

void sort(int arr[],int low,int high){

if (low<high){

int partindx;

partindx=part(arr,low,high);

sort(arr,low,partindx-1);

sort(arr,partindx+1,high);

}

}

int main(){

int size;

int \*arr;

printf("How many elements are to be sorted?");

scanf("%d",&size);

arr=(int \*)malloc(size\*sizeof(int));

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

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

}

printf("Given data:");

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

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

}

sort(arr,0,size);

printf("Sorted data:");

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

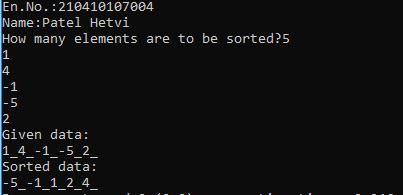
printf("%d\_",arr[i]);

}

return 0;

}

Output:



**13. Program to implement Merge Sort.**

Code:

#include <stdio.h>

#include <stdlib.h>

//MS-Merge sort

//merge function for merging the list.

void merge(int arr[],int low,int mid,int high){

    int s1=mid-low+1;

    int s2=high-mid;

    int lindx=0,rindx=0;

    int indx=low;// array indx

    int \*larr,\*rarr;

    larr=(int\*)malloc(sizeof(int)\*s1);

    rarr=(int\*)malloc(sizeof(int)\*s2);

    // copy the elements in the new array;

    for(int lindx=0;lindx<s1;lindx++){

        larr[lindx]=arr[low+lindx];

    }

    for(int rindx=0;rindx<s2;rindx++){

        rarr[rindx]=arr[mid+rindx+1];

    }

    //compare and merge;

    while(lindx<s1 && rindx<s2){

        if(larr[lindx]<=rarr[rindx]){

            arr[indx]=larr[lindx];

            lindx++;

        }

        else{

            arr[indx]=rarr[rindx];

            rindx++;

        }

        indx++;

    }

    //dump if any elements

    while(lindx<s1){

        arr[indx]=larr[lindx];

        lindx++;

        indx++;

    }

    while(rindx<s2){

        arr[indx]=rarr[rindx];

        rindx++;

        indx++;

    }

}

void msort(int arr[],int low,int high){

    if(low<high){

    int mid;

    mid=(low+high)/2;

    msort(arr,low,mid);

    msort(arr,mid+1,high);

    merge(arr,low,mid,high);

    }

}

void disp(int arr[],int size){

    for(int i=0;i<size;i++){

        printf("%d\_",arr[i]);

    }

    printf("\n");

}

int main()

{

    printf("En.No.:210410107004\nName:Patel Hetvi\n");

    int size;

    printf("Enter the size of array:");

    scanf("%d",&size);

    int \*arr;

    arr=(int\*)malloc(sizeof(int)\*size);

    for(int i=0;i<size;i++){

        scanf("%d",&arr[i]);

    }

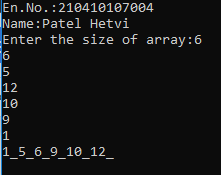
    msort(arr,0,size-1);

    disp(arr,size);

    return 0;

}

Output:



**14. Program to implement linear search.**

Code:

#include <stdio.h>

#include <stdlib.h>

int lsearch(int arr[],int element,int len){

    for(int j=0;j<len;j++){

        if(arr[j]==element){

            return j;

        }

    }

    return -1;

}

int main()

{   printf("En.No.:210410107004\nName:Patel Hetvi\n");

    int MS;

    printf("how many elements are there ?");

    scanf("%d",&MS);

    int \*arr;

    arr=(int\*)malloc(sizeof(int)\*MS);

    printf("Enter array elements:\n");

    for (int i=0;i<MS;i++){

        scanf("%d",&arr[i]);

    }

    int ele,indx;

    //get the element to be searched.

    printf("Enter the element to be searched:");

    scanf("%d",&ele);

    indx=lsearch(arr,ele,MS);

    if(indx!=-1){

        printf("%d found at %d",ele,indx);

    }

    else{

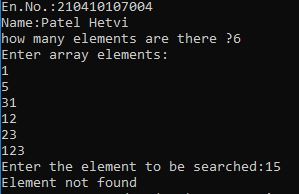
        printf("Element not found");

    }

    return 0;

}

Output:



**15. Program to implement binary search.**

Code:

#include <stdio.h>

#include <stdlib.h>

int binsearch(int arr[],int element,int low,int high){

    int middle;

    while(low<=high){

        middle=(low+high)/2;

    if( element==arr[middle]){

        return middle;

    }

    else if(element<arr[middle]){

        high=middle-1;

    }

    else{

        low=middle+1;

    }

    }

return -1;

}

int binsearch\_recur(int arr[],int element,int low,int high){

    if(low>high){

        return -1;

    }

    int middle=(low+high)/2;

    if( element==arr[middle]){

        return middle;

    }

    else if(element<arr[middle]){

        binsearch\_recur(arr,element,low,middle-1);

    }

    else{

        binsearch\_recur(arr,element,middle+1,high);

    }

}

int main()

{   printf("En.No.:210410107004\nName:Patel Hetvi\n");

    int MS;

    printf("how many elements are there ?");

    scanf("%d",&MS);

    int \*arr;

    arr=(int\*)malloc(sizeof(int)\*MS);

    printf("Enter array elements(sorted):\n");

    for (int i=0;i<MS;i++){

        scanf("%d",&arr[i]);

    }

    int ele,indx,indx2;

    //get the element to be searched.

    printf("Enter the element to be searched:");

    scanf("%d",&ele);

    printf("Searching element using iterative method\n");

    indx=binsearch(arr,ele,0,MS);

    if(indx!=-1){

        printf("%d found at %d\n",ele,indx);

    }

    else{

        printf("Element not found!!!!\n");

    }

    printf("Searching element using recursive method\n");

    indx2=binsearch\_recur(arr,ele,0,MS-1);

    if(indx2!=-1){

        printf("%d found at %d\n",ele,indx2);

    }

    else{

        printf("!!!!!Element not found");

    }

    return 0;

}

Output:

