

**Data Structure lab Programs**

|  |  |  |
| --- | --- | --- |
| **WEEK** | **TOPIC** | **PGNO** |
|  |  |  |
| 1 | Struct | 02 - 07 |
| 2 | Stack Operations | 08 - 15 |
| 3 | Queue Operations | 16 - 23 |
| 4 | Linked list | 24 - 31 (SLL) & 32 – 39 (DLL) |
| 5 | Circular Linked List | 40 - 49 |
| 6 | Insertion , Bubble,  Optimized Bubble &  Selection Sortings | 50 - 55 |
| 7 | Quick, Heap & Merge | 56 - 58 |
| 8 | Binary Tree | 59 – 61 |
| 9 | Binary Search Tree | 62-69 |
| 10 | AVL Tree | 70-82 |
| 11 | BST&DST | 83 – 84 |
| 12 | KMT & Brute | 85 - 87 |
|  |  |  |
|  |  |  |

**WEEK - 1**

**1.Write a C program to read name, Roll number, year and marks of three subjects of a student and print the student the name, roll number, average marks of the student using structures.**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct Student{

char name[10];

char roll[10];

int year;

float marks[3];

};

int main(){

struct Student s;float avg=0.0;

printf("Enter name: ");

scanf("%s",&s.name);

printf("Enter rollno: ");

scanf("%s",&s.roll);

printf("Enter year: ");

scanf("%d",&s.year);

for(int i=0;i<3;i++){

printf("Enter subject - %d marks",i+1);

scanf("%f",&s.marks[i]);

avg=avg +s.marks[i];

}

printf("\*\*\*\*\*\*\*\*STUDENT DETAILS\*\*\*\*\*\*\*\n");

printf("NAME : %s\n",s.name);

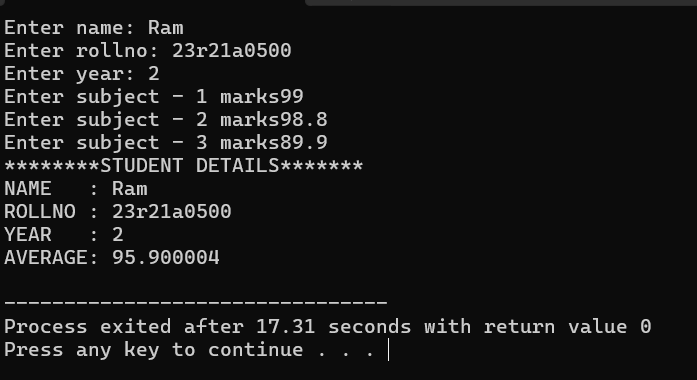
printf("ROLLNO : %s\n",s.roll);

printf("YEAR : %d\n",s.year);

printf("AVERAGE: %f\n",avg/3.0);

return 0;

}**OUTPUT:**

****

**2.Write a c program to read the details of employee(emp no,name,basic salary,HRA,DA) of ‘n’ employees. Calculate the gross salary of each employee and print emp name,no,gross salary.**

**SOURCE CODE:**

#include<stdio.h>

struct Employee{

char name[50];

int eno;

float bs,hra,da,gs;

};

int main()

{

int n,i;

printf("No of employees:");

scanf("%d",&n);

struct Employee e[n];

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

{

printf("enter name: ");

scanf("%s",e[i].name);

printf("enter number: ");

scanf("%d",&e[i].eno);

printf("enter bs: ");

scanf("%f",&e[i].bs);

printf("enter hra: ");

scanf("%f",&e[i].hra);

printf("enter da: ");

scanf("%f",&e[i].da);

e[i].gs=e[i].bs+e[i].hra+e[i].da;

}

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

{

printf("\nEmployee details:%d\n",i+1);

printf("Name:%s\n",e[i].name);

printf("Number:%d\n",e[i].eno);

printf("Gross salary:%.2f",e[i].gs);

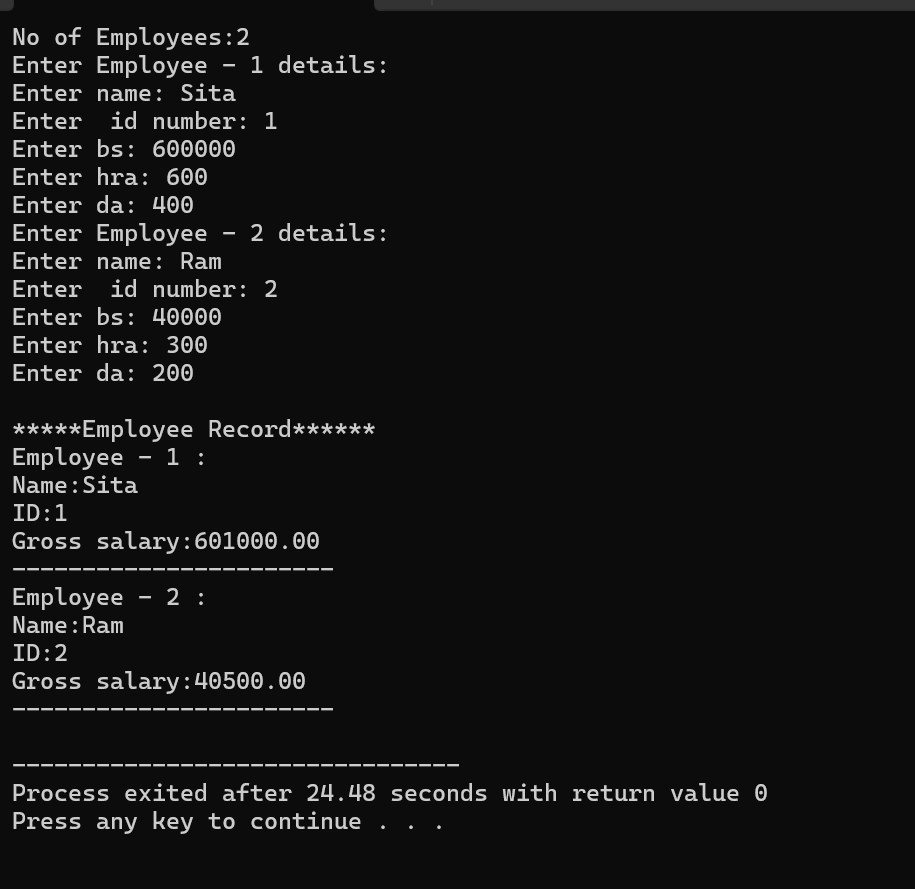
printf("\n-----------------------\n");

}

return 0;

}

**OUTPUT:**



**3.Write a C Program to read student name, roll number and average marks of a student using structure and within another structure read date of birth of a student as day, month and year. Print the student’s name, roll number, day of birth and average marks of a student.**

**Hint: use Nested Structure**

**Source code:**

#include<stdio.h>

struct student {

char name[20];

char roll\_number[10];

float average;

struct address {

int day;

int month;

int year;

} addr;

};

int main() {

struct student s;

printf("Enter student name: ");

scanf("%s",s.name);

printf("Enter student roll number: ");

scanf("%s",&s.roll\_number);

printf("Enter student average marks: ");

scanf("%f",&s.average);

printf("Enter date, month,and year: ");

scanf("%d%d%d",&s.addr.day,&s.addr.month,&s.addr.year);

printf("\nStudent's Details:\n");

printf("Name: %s\n", s.name);

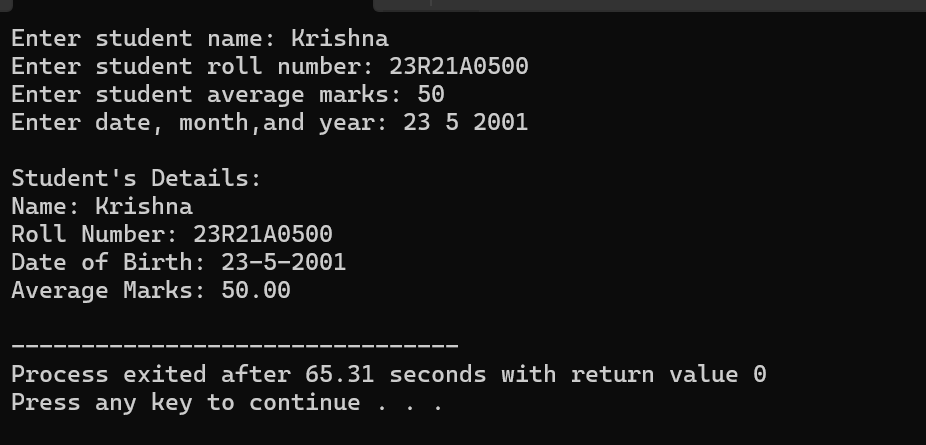
printf("Roll Number: %s\n", s.roll\_number);

printf("Date of Birth: %d-%d-%d\n", s.addr.day,s.addr.month,s.addr.year);

printf("Average Marks: %.2f\n", s.average);

return 0; }

**OUTPUT:**



**4.Write a C program to store name, roll number, year and marks of three subjects of n students and print the student the name, roll number, average and grade based using pointer variable.Hint: use Pointer to Structures**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct Student{

char name[50];

int roll\_number;

int year;

float marks[3];

};

void printStudentInfo(struct Student \*s) {

float total = s->marks[0] + s->marks[1] + s->marks[2];

float average = total / 3.0;

char grade;

if (average >= 90) grade = 'A';

else if (average >= 80) grade = 'B';

else if (average >= 70) grade = 'C';

else if (average >= 60) grade = 'D';

else grade = 'F';

printf("Name: %s\n", s->name);

printf("Roll Number: %d\n", s->roll\_number);

printf("Average Marks: %.2f\n", average);

printf("Grade: %c\n", grade);

}

int main() {

int n;

printf("Enter the number of students: ");

scanf("%d", &n);

struct Student \*students = (struct Student \*)malloc(n \* sizeof(struct Student));

if (students == NULL) {

printf("Memory allocation failed\n");

return 1;

}

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

printf("Enter details for student %d:\n", i + 1);

printf("Name: ");

scanf("%s", students[i].name);

printf("Roll Number: ");

scanf("%d", &students[i].roll\_number);

printf("Year: ");

scanf("%d", &students[i].year);

printf("Enter marks for 3 subjects: ");

for (int j = 0; j < 3; j++) {

scanf("%f", &students[i].marks[j]);

}

}

printf("\nStudent Information:\n");

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

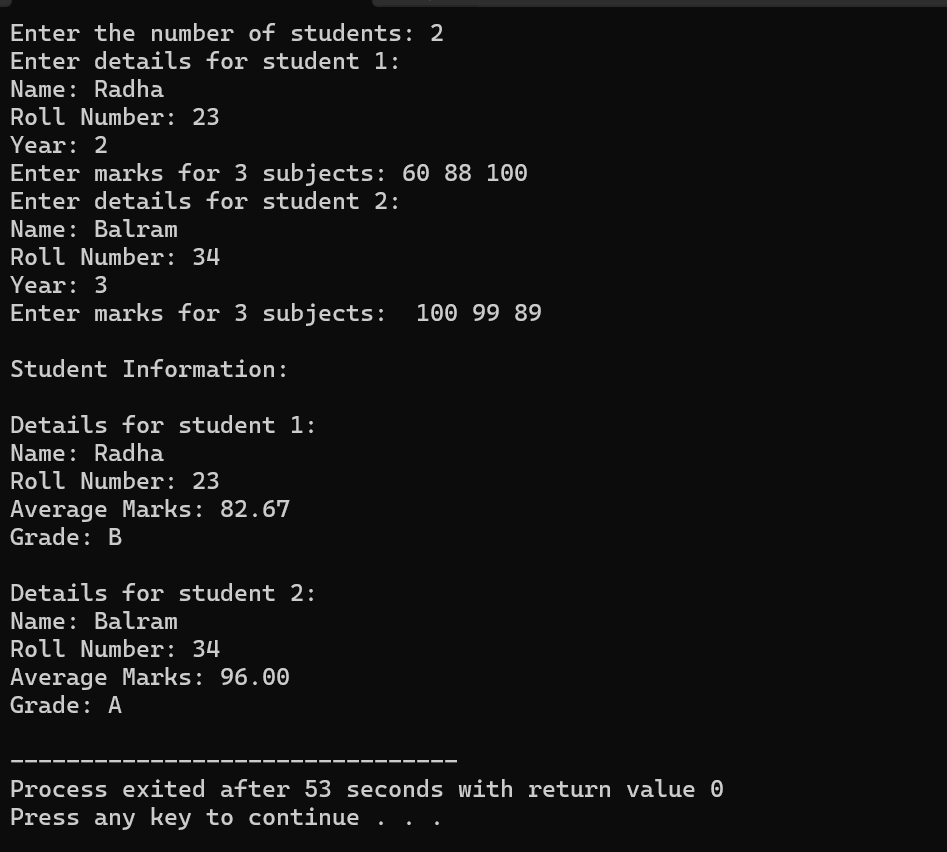
printf("\nDetails for student %d:\n", i + 1);

printStudentInfo(&students[i]);

}

free(students); return 0;}

**OUTPUT:**



**WEEK-2**

**1.Write a C Program to Implement a Stack using Array and perform**

**Insert integers into the stack**

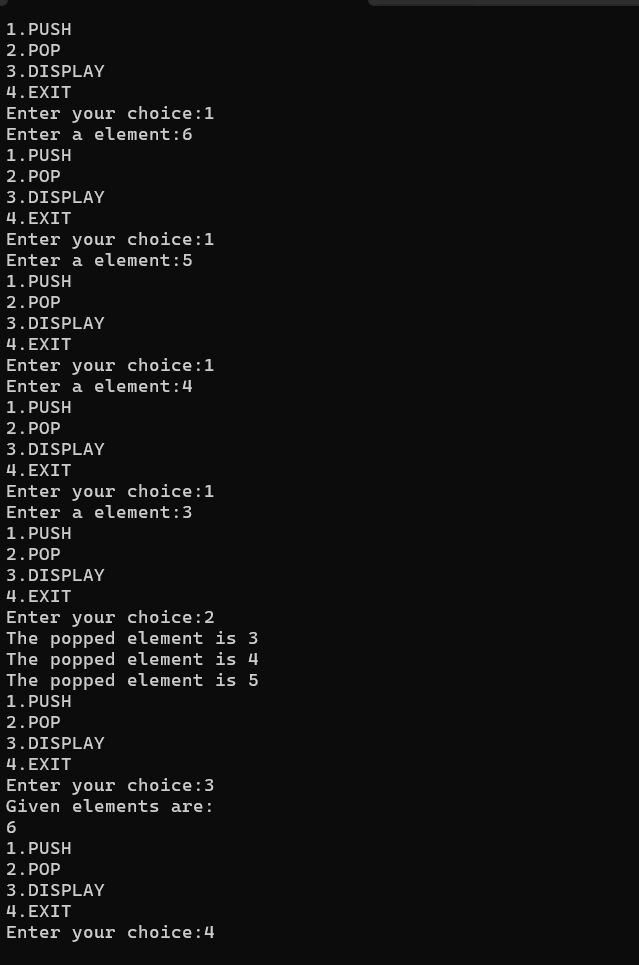
**Delete three elements from stack**

**Display the stack in reverse order**

**SOURCE CODE:**

#include<stdio.h>  
#include<stdlib.h>  
#define N 6  
int a[N],top=-1;  
void push(){  
if(top==N-1){  
printf("overflow..");}  
else{  
top++;  
printf("enter a element:");  
scanf("%d",&a[top]);}}  
void pop(){  
if(top==-1){  
printf("underflow..\n");}  
else{  
printf("the popped element is %d \n",a[top]);  
top--;}  
void display(){  
printf("Given elements are:\n");  
for( int i=top;i>=0;i--){  
printf("%d \n",a[i]);}}  
int main(){  
int choice=0;  
do{  
printf("1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT \n");  
printf("enter your choice:");  
scanf("%d",&choice);  
switch (choice){  
case 1:push();  
      break;  
case 2: pop();  
       pop();  
       pop();  
       break;  
   case 3:display();  
          break;  
   case 4: exit(0);  
default : printf("INVALID CHOICE...");}}while (choice !=4);  
return 0;}

**OUTPUT:**



**2.Write a C Program to Implement a Stack using Array and perform**

**Insert characters into the stack**

**Delete three elements from stack**

**Display the stack in reverse order**

**For example, if stack elements are A B C D E F, then the function should print C B A after removing F, E, D**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#define N 6

char a[N],top=-1;

int i; void push(){

if(top==N-1)

printf("Overflow...\n");

else{

top++;

printf("Enter a character...");

scanf("%s",&a[top]);

} }

void pop(){

if(top==-1)

printf("Underflow...\n");

else{

printf("Popped element is %c\n",a[top]);

top--; } }

void display(){

printf("Stack elements are :\n");

for(i=top;i>=0;i--){

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

int main() {

int ch;

do {

printf("1:PUSH\n2:POP\n3:DISPLAY\n4:EXIT\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch) {

case 1:push();

push();

push();

push();

push();

push();

break;

case 2:pop();

pop();

pop();

break;

case 3:display();

break;

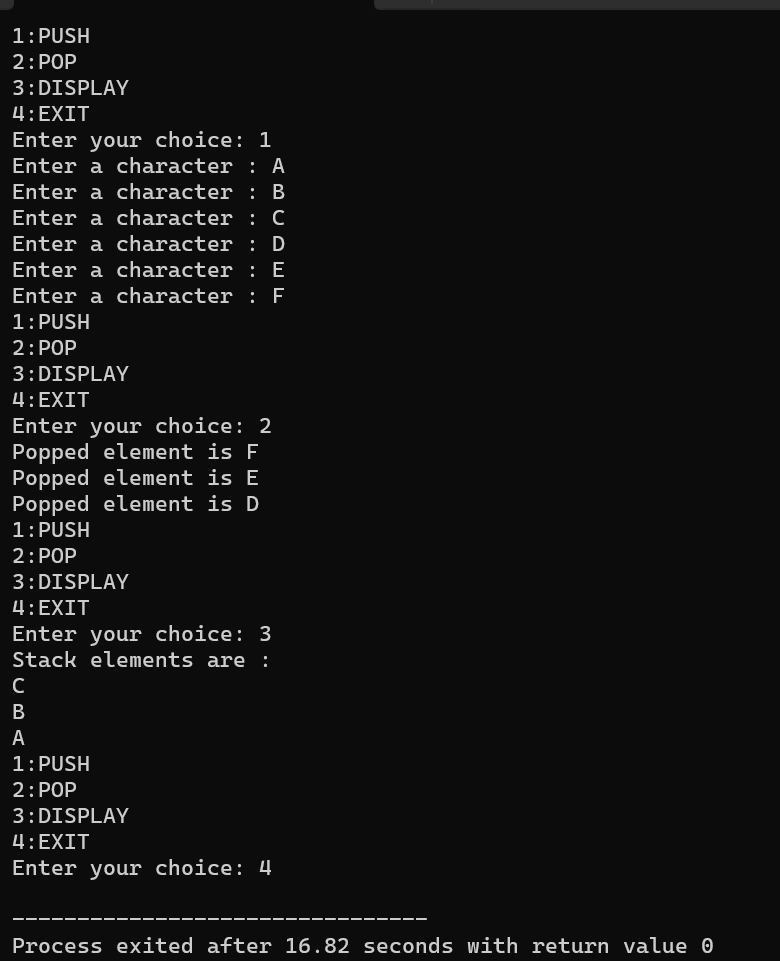
case 4: break;

default:printf("Invalid choice...\n"); } }

while(ch!=4);

return 0; }

**OUTPUT:**



**3.Write a C program to check string is palindrome using stack operations**

**for example, if string is “MADAM”, the function should print**

**“PALINDROME” otherwise “NOT a PALINDROME**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#define N 5

char a[N],top=-1;

void push(){

if(top==N-1){

printf("overflow..");

}

else{

top++;

printf("enter a element:");

scanf("%s",&a[top]);

}

}

void display(){

int i;

printf("Given elements are:\n");

for( i=top;i>=0;i--){

printf("%c\t",a[i]);

}

}

int main(){

int choice=0 ,i;

do{

printf("1.Push\n2.Exit\n3.Display\n");

printf("enter your choice");

scanf("%d",&choice);

if (choice ==1){

push();

}

else if (choice ==3){

display();

}

}while (choice !=2);

char b[N];

for(i=top;i>=0;i--){

for(int j=0;j>N-1;j++){

b[j]=a[i];

}

}

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

if(a[i]!=b[i]){

printf("NOT PALINDROME..");

exit(0);

}

else {

printf("PALINDROME..");

exit(0);

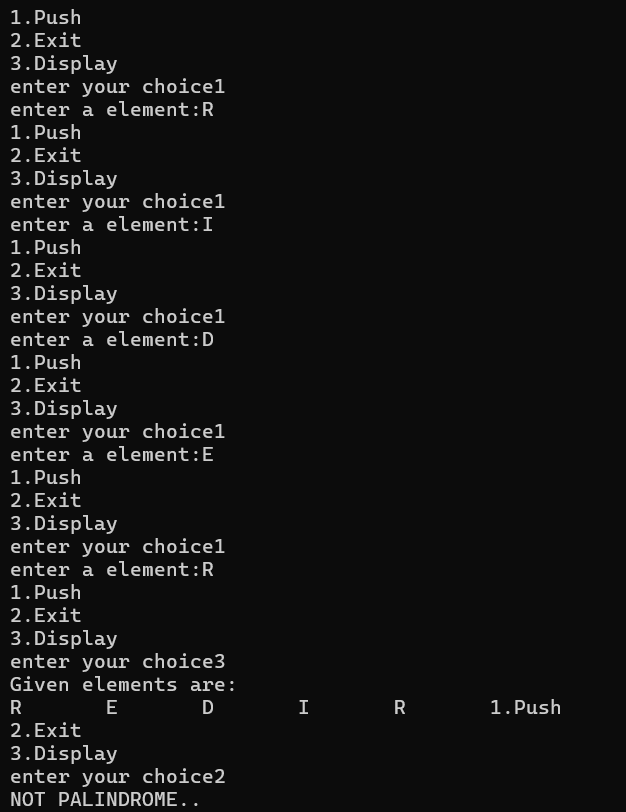
}

}

return 0;

}

**OUTPUT:**

****

**4.Write a C Program to Implement a Stack using Array and perform all operations**.

**SOURCE CODE:**

#include<stdio.h>

#define N 5

int a[N],top=-1,i;

void push(){

if(top==N-1)

printf("Overflow...\n");

else{ top++;

printf("Enter a value...");

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

}

}

void pop(){

if(top==-1)

printf("Underflow...\n");

else{

printf("popped element is %d\n",a[top]);

top--;

}

}

void display(){

printf("Stack elements are:\n");

for(i=top;i>=0;i--)

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

int main() {

int ch;

do {

printf("1:push\n2:pop\n3:display\n4:exit\n");

printf("enter your choice..");

scanf("%d",&ch);

switch(ch){

case 1:push();

break;

case 2:pop();

break;

case 3:display();

break;

case 4:break;

default:

printf("invalid choice..\n");

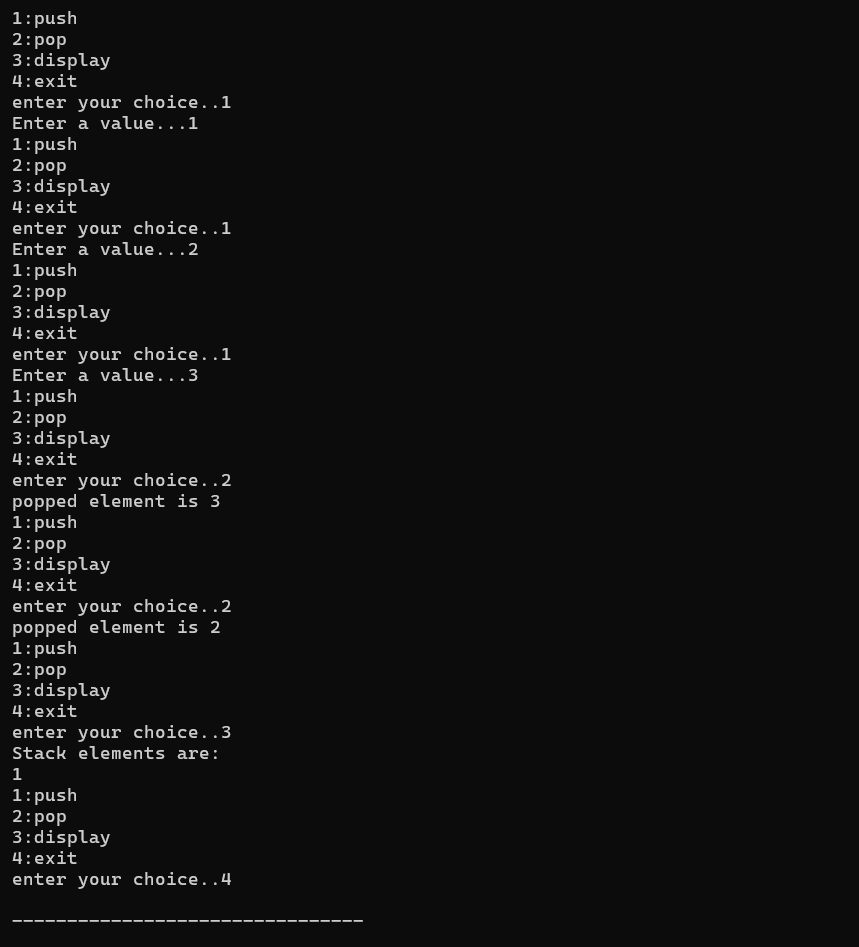
}

}

while(ch!=4);

return 0; }

**OUTPUT:**



**WEEK-3**

**1.Write a C Program to Implement a Queue using Array and perform**

**Insert integers into the Queue**

**Delete three elements from Queue**

**Display the Queue Elements**

**SOURCE CODE:**

#include<stdio.h>

#define N 6

int a[N],fr=-1,rr=-1;

void enqueue(){

if(rr==N-1)

printf("Overflow...\n");

else{

if(fr==-1 & rr==-1)

fr++;

rr++;

printf("Enter a value: ");

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

}

}

void dequeue(){

if(fr==-1 & rr==-1)

printf("Underflow...\n");

else{

printf("Dequeued value is %d\n",a[fr]);

fr++;

if(fr>rr)

fr=rr=-1;

}

}

void display(){

int i;

if(fr==-1 & rr==-1){

printf("Queue is empty...\n");

}

else{

printf("Queue elements are:\n");

for(i=fr;i<=rr;i++)

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

}

}

int main(){

int ch;

do {

printf("1:Enqueue\n2:Dequeue\n3:Display\n4:Exit\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch){

case 1:enqueue();

break;

case 2:dequeue();

dequeue();

dequeue();

break;

case 3:display();

break;

case 4:break;

default:

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

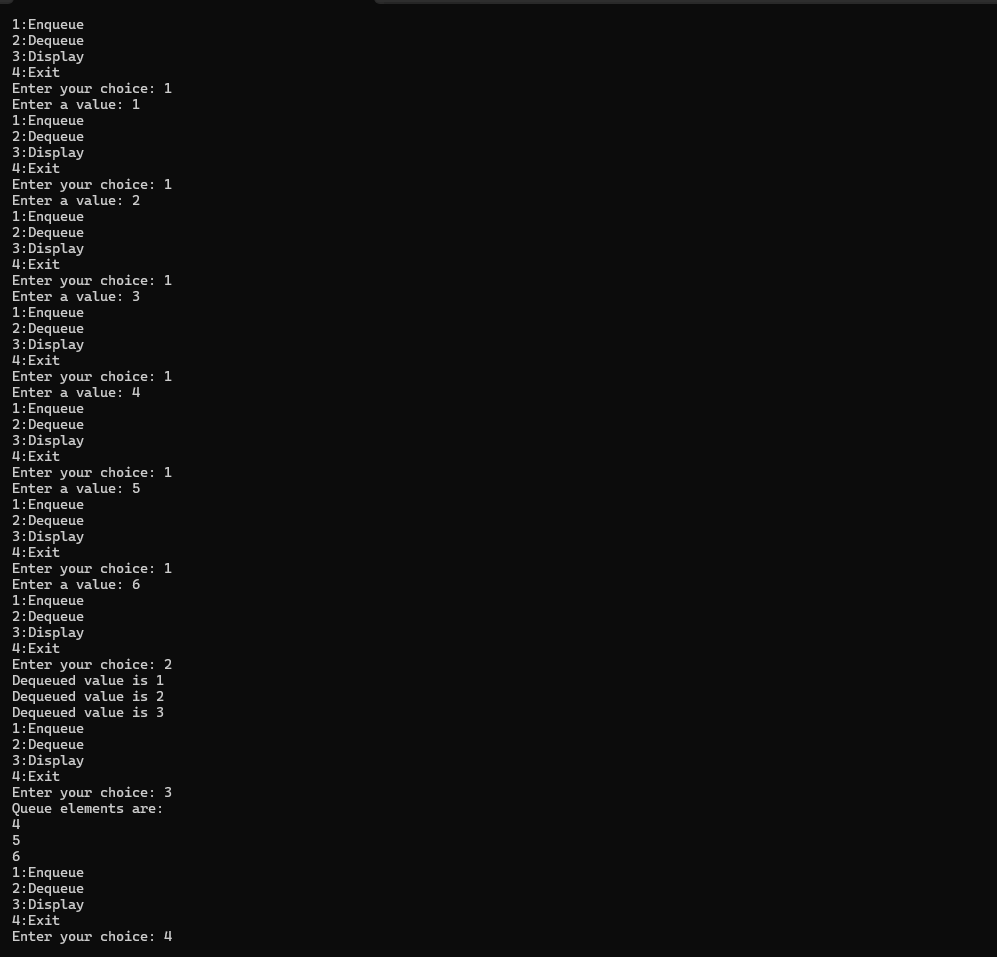
}

}while(ch!=4);

return 0;

}

**OUTPUT:**



**2 Write a C Program to Implement a Queue using Array and perform**

**• Insert characters into the Queue**

**• Delete three elements from Queue**

**• Display the Queue Elements**

**For example, if Queue elements are A B C D E F then the function should print D E F after removing A B C**

**SOURCE CODE:**

#include<stdio.h>

#define N 6

char a[N];

int fr=-1,rr=-1;

void Enqueue(){ if (rr==N-1)

printf("Overflow...");

else{

if(fr==-1 & rr==-1)

fr++; rr++;

printf("Enter your element:");

scanf("%s",&a[rr]); }}

void Dequeue(){if (fr==-1 & rr==-1)

printf("underflow...");

else{ printf("The dequeues element is %c\n",a[fr]);

fr++;

if(fr>rr) fr=rr=-1;}}

void Display(){int i;

if (fr==-1&rr==-1)

printf("Queue is empty");

else{ printf("Queue elements are:");

for(i=fr;i<=rr;i++)

printf("%c\t",a[i]);}

printf("\n");}

int main(){int ch;

do{

printf("1.Enqueue \n2.Dequeue \n3.Display \n4.Exit\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch){

case 1:Enqueue();

Enqueue();

Enqueue();

Enqueue();

Enqueue();

Enqueue();

break;

case 2:

Dequeue();

Dequeue();

Dequeue();

break;

case 3:

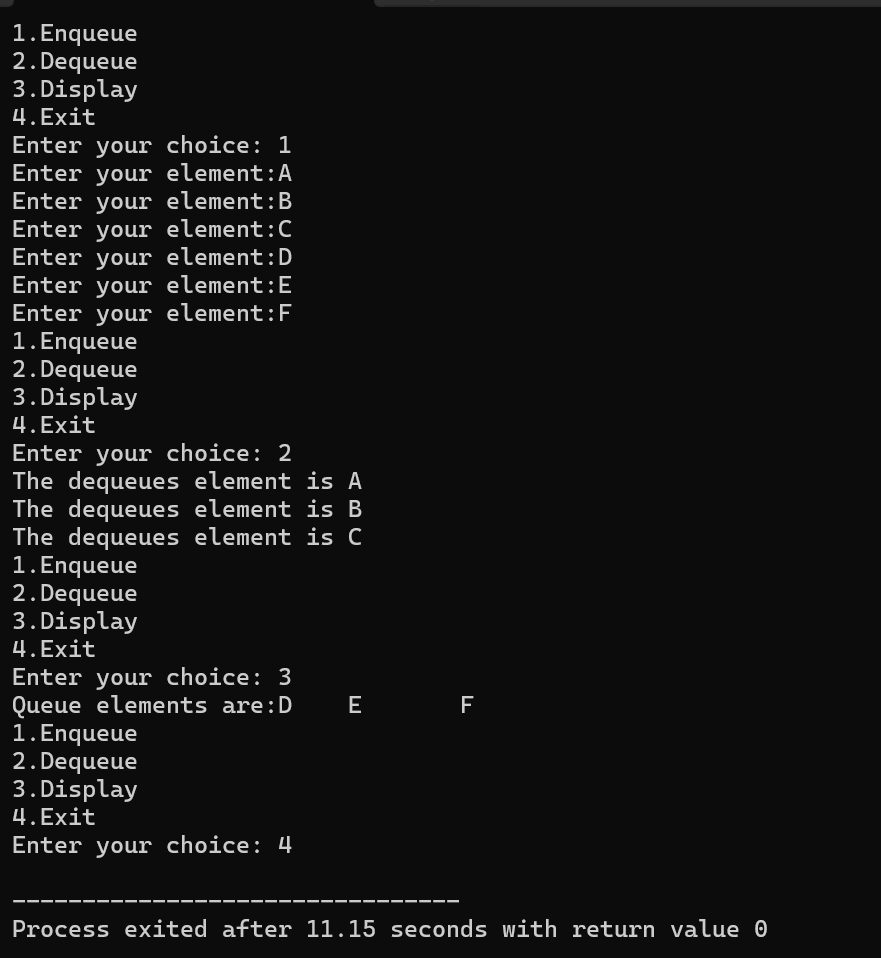
Display();

break;

} }while(ch!=4);

return 0;}

**OUTPUT:**



**3 Write a C Program to Implement a Queue using Array and perform**

**• Insert integers into the Queue**

**• Search the elements from Queue**

**SOURCE CODE:**

#include<stdio.h>

#define N 6

int a[N],fr=-1,rr=-1;

void enqueue(){

if(rr==N-1)

printf("Overflow...\n");

else{

if(fr==-1 & rr==-1)

fr++;

rr++;

printf("Enter a value: ");

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

}

}

void search(){

int i,key,flag=0;

printf("Enter your key to search: ");

scanf("%d",&key);

for(i=fr;i<=rr;i++){

if(a[i]==key){

flag=1;

break;

}

}

if(flag==1)

printf("Key found...\n");

else

printf("No such key found...\n");

}

void display(){

int i;

if(fr==-1 & rr==-1){

printf("Queue is empty...\n");

}

else{

printf("Queue elements are:\n");

for(i=fr;i<=rr;i++)

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

}

}

int main(){

int ch;

do {

printf("1:Enqueue\n2:Search\n3:Display\n4:Exit\n");

printf("Enter your choice: ");

scanf("%d",&ch);

switch(ch){

case 1:

enqueue();

break;

case 2:

search();

break;

case 3:

display();

break;

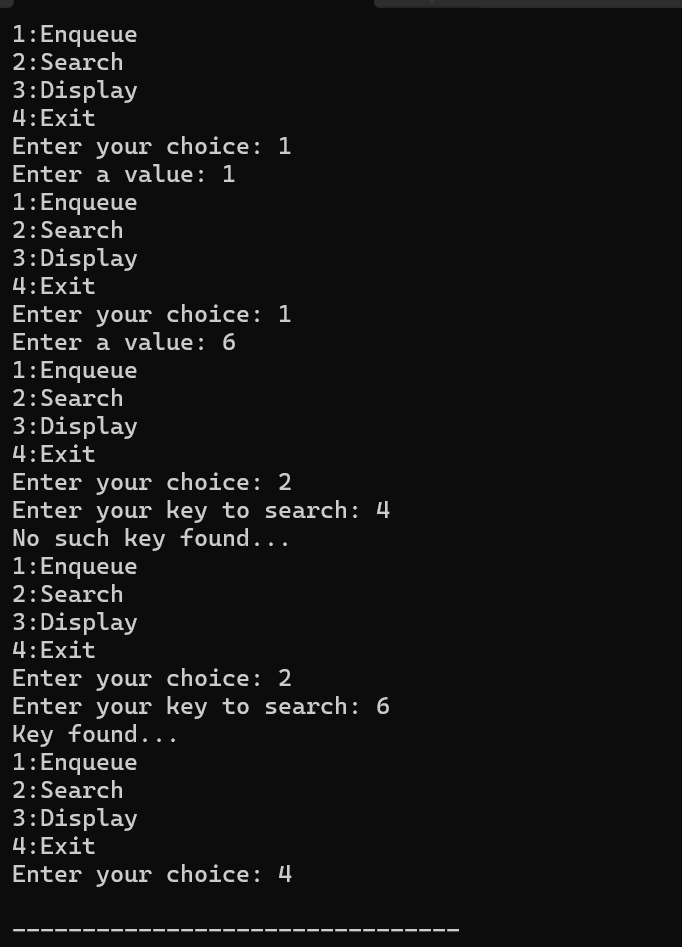
case 4:

break;

} }while(ch!=4);

return 0; }

**OUTPUT:**



**4 Write a C program to implement a Queue using array and perform all the operations.**

#include<stdio.h>

#define N 6

int a[N],fr=-1,rr=-1;

void Enqueue(){

if (rr==N-1)

printf("Overflow...");

else{

if(fr==-1 & rr==-1)

fr++;

rr++;

printf("Enter your element:");

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

void Dequeue(){

if (fr==-1 & rr==-1)

printf("underflow...");

else{

printf("The dequeues element is %d\n",a[fr]);

fr++;

if(fr>rr)

fr=rr=-1;}}

void Display(){

int i;

if (fr==-1&rr==-1)

printf("Queue is empty");

else{ printf("Queue elements are:");

for(i=fr;i<=rr;i++)

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

int main(){int ch;

do{

printf("\n1.Enqueue \n 2.Dequeue \n 3.Display \n 4.Exit\n");

printf("Enter your choice:");

scanf("%d",&ch);

switch(ch){

case 1:Enqueue();

break;

case 2:

Dequeue();

break;

case 3:

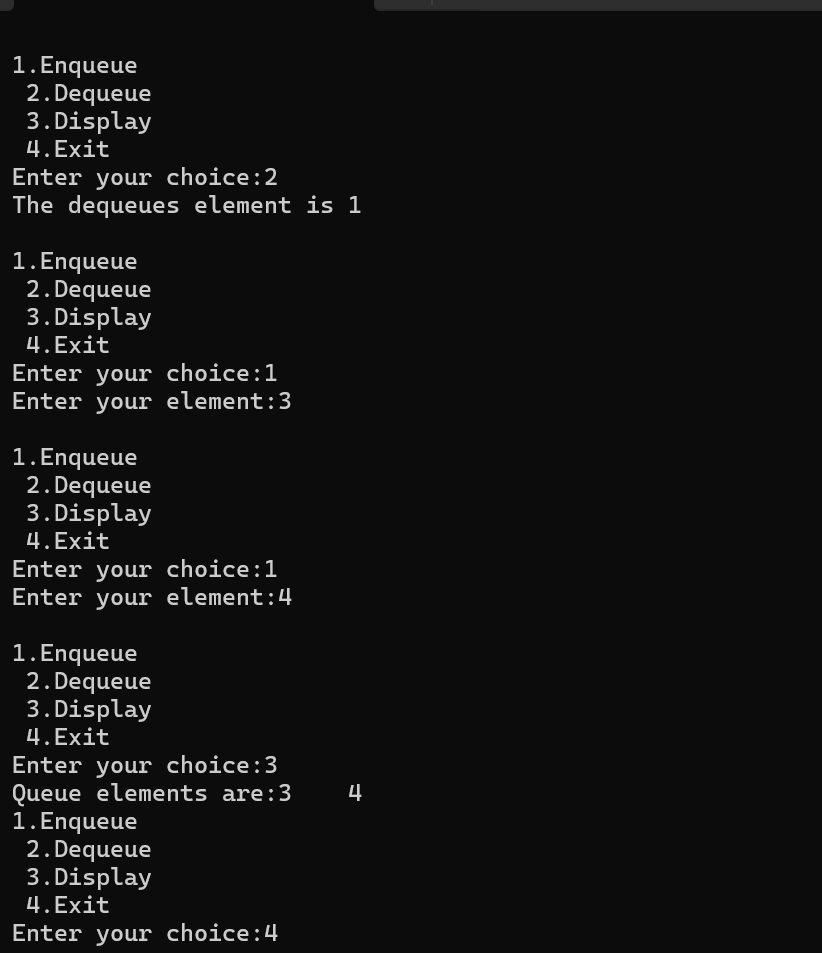
Display();

break;

default:printf("invalid choice.."); } }while(ch!=4);

return 0;}

**OUTPUT:**



**Week-4**

**PROBLEM STATEMENT :**

1.Write a C program that uses functions to perform the following:

a. Create a Single linked list of integers.

b. Insert an Element at a Position

c. Display the contents of the Single linked list.

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*nt;

}\*st=NULL,\*tp=NULL,\*nn=NULL;

void display(){

tp=st;

printf("Singly linked list elements are:");

while(tp!=NULL){

printf("%d\t",tp->dt);

tp=tp->nt;

}

}

void create(){

int ch;

do{

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

printf("Enter data:");

scanf("%d",&nn->dt);

if(st==NULL)

st=tp=nn;

else {

tp->nt=nn;

tp=tp->nt;

}

printf("0 for exit other to continue:");

scanf("%d",&ch);

}while(ch!=0);

tp->nt=NULL;

}

void insertion\_at\_position(){ int ref,i;

printf("\nEnter a position :");

scanf("%d",&ref);

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

printf("Enter data to add at %d position:",ref);

scanf("%d",&nn->dt);

tp=st;

for(i=1;i<(ref-1);i++){

tp=tp->nt;

}nn->nt=tp->nt;

tp->nt=nn;

}

int main(){

create();

display();

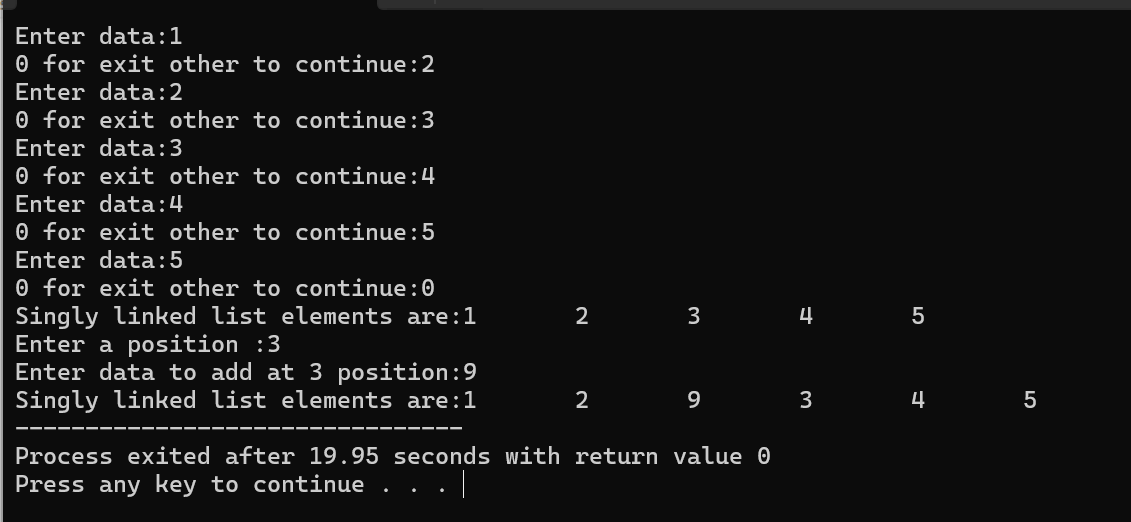
insertion\_at\_position();

display();

return 0;

}

**OUTPUT:**



**PROBLEM STATEMENT :**

2.Write a C program that uses functions to perform the following:

a. Create a Single linked list of characters.

b. Insert Element in a Single linked list at begin, Last

c. Display the contents of the Single linked.

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*nt;

}\*st=NULL,\*tp=NULL,\*nn=NULL;

void display(){

tp=st;

printf("Singly linked list elements are:");

while(tp!=NULL){

printf("%d\t",tp->dt);

tp=tp->nt;}}

void create(){

int ch;

do{

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

printf("Enter data:");

scanf("%d",&nn->dt);

if(st==NULL)

st=tp=nn;

else {tp->nt=nn;

tp=tp->nt;}

printf("0 for exit other to continue:");

scanf("%d",&ch);

}while(ch!=0);

tp->nt=NULL;}

void insertion\_at\_begin(){

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

printf("\nEnter data to insert at begin:");

scanf("%d",&nn->dt);

nn->nt=st;

st=nn;}

void insertion\_at\_end(){

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

printf("\nEnter data to insert at end:");

scanf("%d",&nn->dt);

tp=st;

while(tp->nt!=NULL)

tp=tp->nt;

tp->nt=nn;

nn->nt=NULL;}

int main(){

create();

display();

insertion\_at\_begin();

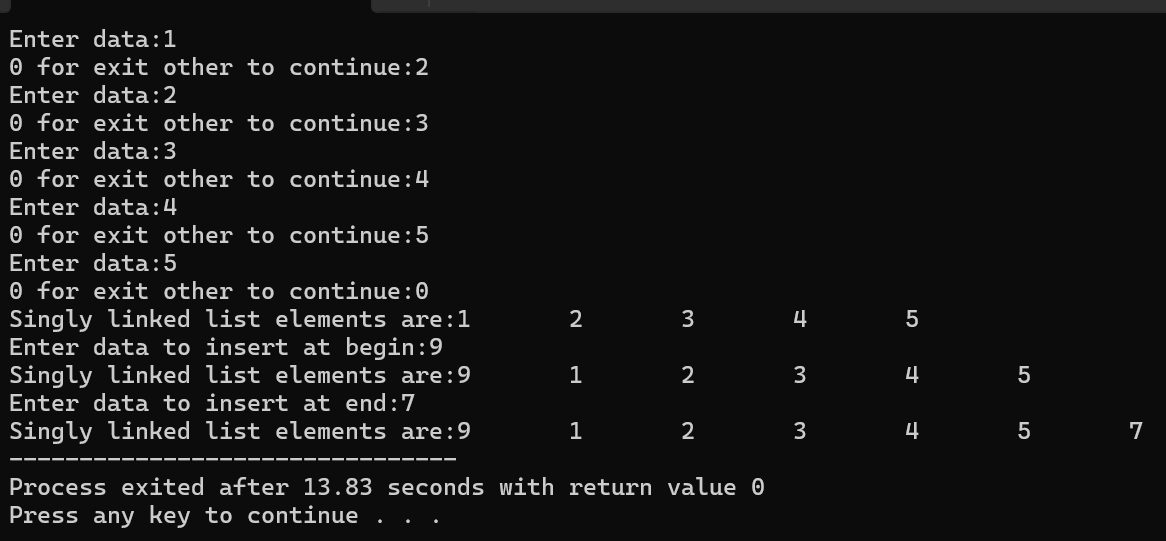
display();

insertion\_at\_end();

display();

return 0;}

**OUTPUT:**



**PROBLEM STATEMENT :**

3.Write a C program that uses functions to perform the following:

a. Create a Single linked listartart of integers.

b. Display the Nodes

c. Delete a node at Beginning and delete a node after a given element

**SOURCE CODE:**

/\*3.Write a C program that uses functions to perform the following:

a. Create a Single linked listartart of integers.

b. Display the Nodes

c. Delete a node at Beginning and after \*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*nt;

}\*start=NULL,\*temporary=NULL,\*next\_node=NULL;

void display(){

temporary=start;

printf("\nSingly linked list elements are:");

while(temporary!=NULL){

printf("%d\t",temporary->data);

temporary=temporary->nt;

}

}

void delete\_at\_begin(){

temporary=start;

start=temporary->nt;

free(temporary);

}

void Delete\_After\_Given\_Element(){

int ref;

printf("\nEnter after which element to delete : ");

scanf("%d",&ref);

temporary=start;

while(temporary->data!=ref){

temporary=temporary->nt;

}

next\_node=temporary->nt;

temporary->nt=next\_node->nt;

free(next\_node);

}

void create(){

int choice;

do{

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

printf("Enter data:");

scanf("%d",&next\_node->data);

if(start==NULL)

start=temporary=next\_node;

else {

temporary->nt=next\_node;

temporary=temporary->nt;

}

printf("0 for exit other to continue:");

scanf("%d",&choice);

}while(choice!=0);

temporary->nt=NULL;

}

int main(){

create();

display();

delete\_at\_begin();

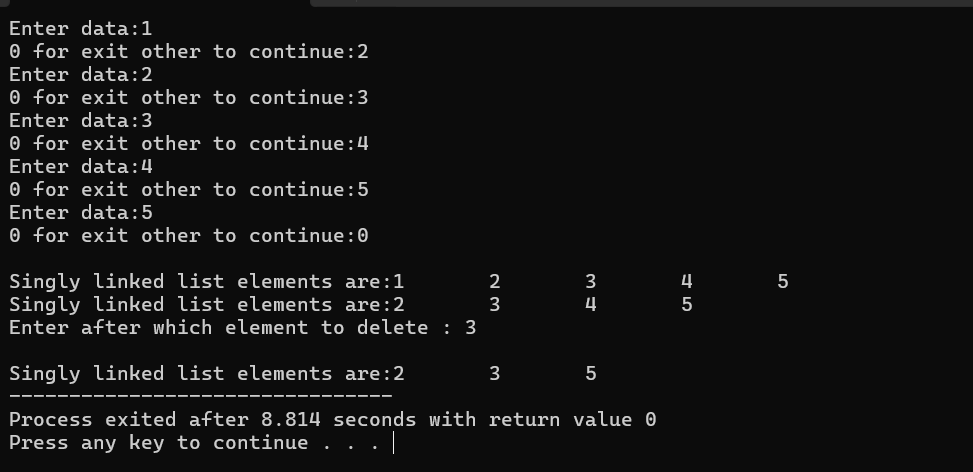
display();

Delete\_After\_Given\_Element();

display();

return 0;

}**OUTPUT:**

****

**PROBLEM STATEMENT :**

4.Write a C program that uses functions to perform the following:

a. Create a Single linked list of integers.

b. Display the Nodes

c. Delete a node at a position

**SOURCE CODE:**

/\*4.Write a C program that uses functions to perform the following:

a. Create a Single linked list of integers.

b. Display the Nodes

c. Delete a node at a position\*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*nt;

}\*start=NULL,\*temporary=NULL,\*next\_node=NULL;

void display(){

temporary=start;

printf("\nSingly linked list elements are:");

while(temporary!=NULL){

printf("%d\t",temporary->data);

temporary=temporary->nt;

}

}

void create(){

int choice;

do{

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

printf("Enter data:");

scanf("%d",&next\_node->data);

if(start==NULL)

start=temporary=next\_node;

else {

temporary->nt=next\_node;

temporary=temporary->nt;

}

printf("0 for exit other to continue:");

scanf("%d",&choice);

}while(choice!=0);

temporary->nt=NULL;

}

void delete\_at\_particular\_position(){

int ref,i;

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

scanf("%d",&ref);

temporary=start;

for(i=1;i<(ref-1);i++)

temporary=temporary->nt;

next\_node=temporary->nt;

temporary->nt=next\_node->nt;

free(next\_node);

}

int main(){

create();

display();

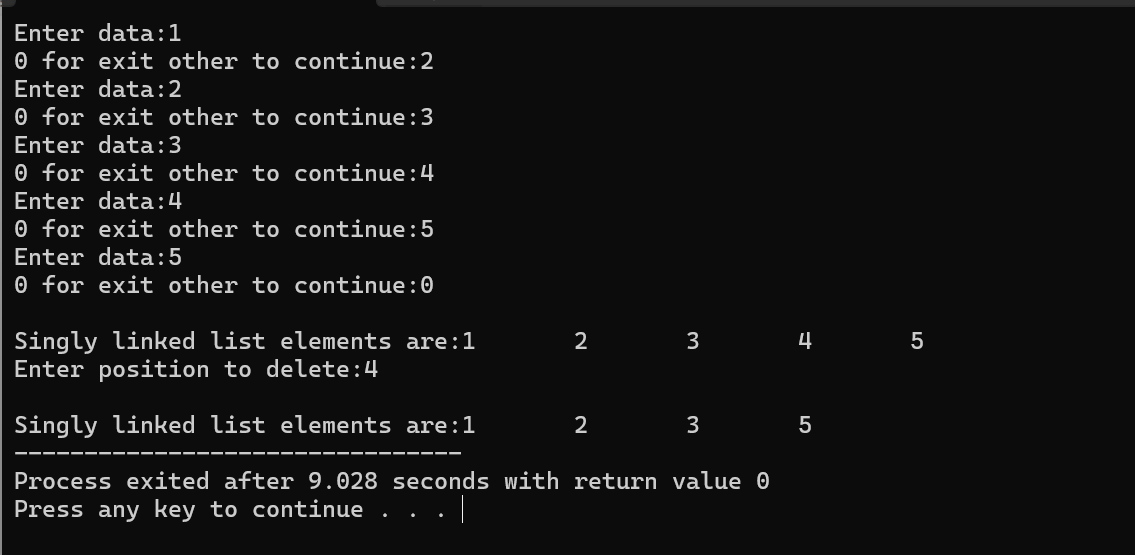
delete\_at\_particular\_position();

display();

return 0;

}

**OUTPUT:**



**PROBLEM STATEMENT :**

5.Write a C program that uses functions to perform the following:

a. Create a Double linked list of integers.

b. Insert Element at begin, Last.

c. Display the contents of the above list.

**SOURCE CODE:**

/\* 5.Write a C program that uses functions to perform the following:

a. Create a Double linked list of integers.

b. Insert Element at begin, Last.

c. Display the contents of the above list.\*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next,\*previous;

}\*start=NULL,\*temporary=NULL,\*new\_node=NULL;

void display(){

temporary=start;

printf("double linked listart elements are:");

while(temporary!=NULL){

printf("%d\t",temporary->data);

temporary=temporary->next;

}

}

void create(){

int choice;

do{

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

printf("Enter data:");

scanf("%d",&new\_node->data);

if(start==NULL){

start=temporary=new\_node;

start->previous=NULL;

}

else {

temporary->next=new\_node;

new\_node->previous=temporary;

temporary=temporary->next;

}

printf("0 for exit other to continue:");

scanf("%d",&choice);

}while(choice!=0);

temporary->next=NULL;

}

void insertion\_at\_begin(){

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

printf("\nEnter data to add at begin:");

scanf("%d",&new\_node->data);

new\_node->next=start;

new\_node->previous=NULL;

start->previous=new\_node;

start=new\_node;

}

void insertion\_at\_end(){

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

printf("\nEnter data to add at end:");

scanf("%d",&new\_node->data);

temporary=start;

while(temporary->next!=NULL)

temporary=temporary->next;

new\_node->previous=temporary;

temporary->next=new\_node;

new\_node->next=NULL;

}

void displayReverse(struct node\* tp) {

if (tp == NULL) {

return; // Base case: if the node is NULL, return

}

// Recursive call to the next node

displayReverse(tp->next);

// Print the current node's data after the recursive call

printf("%d ", tp->data);

} int main(){

create();

display();

insertion\_at\_begin();

display();

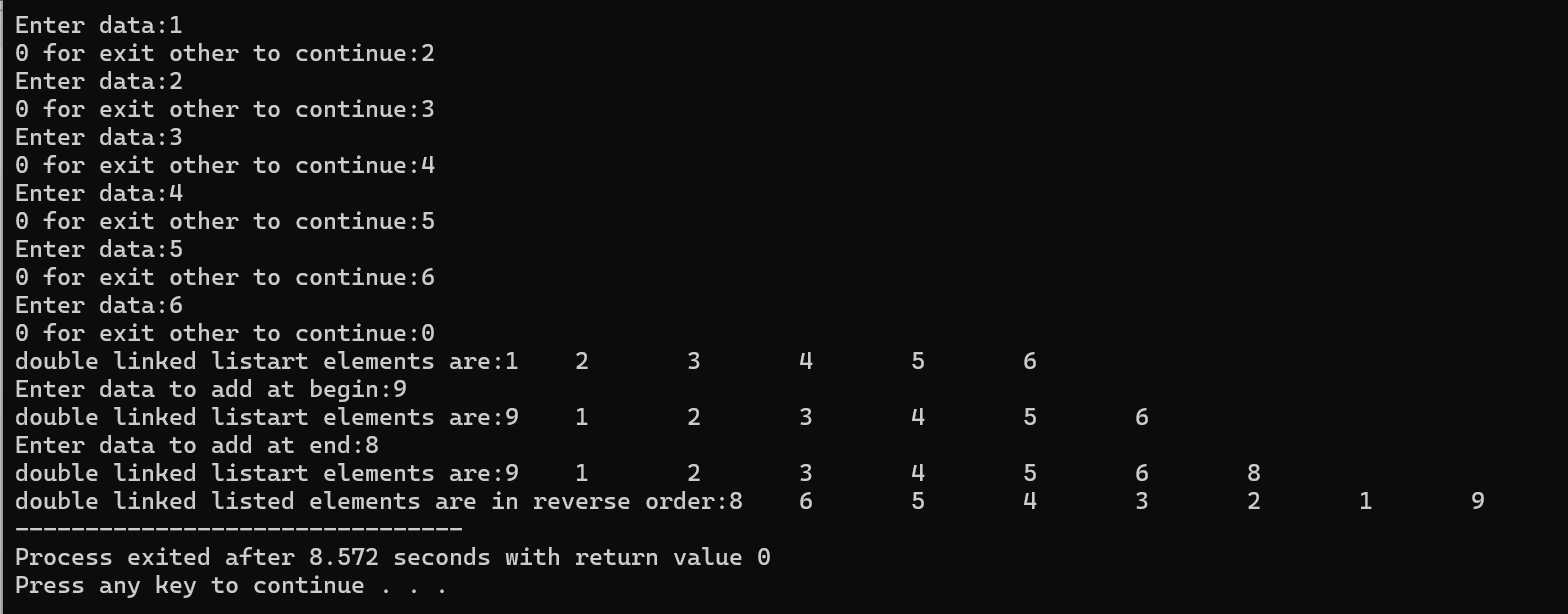
insertion\_at\_end();

display();

displayReverse(start);

return 0;

}**OUTPUT:**



**PROBLEM STATEMENT :**

6.Write a C program that uses functions to perform the following:

a. Create a Double linked list of integers.

b. Insert Element at a position.

c. Display the contents of the above list.

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*nt,\*pr;}\*st=NULL,\*tp=NULL,\*nn=NULL;

void display(){

tp=st;

printf("Double linked list elements are:");

while(tp!=NULL){

printf("%d\t",tp->dt);

tp=tp->nt;}}

void create(){

int ch;

do{

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

printf("Enter data:");

scanf("%d",&nn->dt);

if(st==NULL){

st=tp=nn;

st->pr=NULL;}

else {tp->nt=nn;

nn->pr=tp;

tp=tp->nt;}

printf("0 for exit other to continue:");

scanf("%d",&ch);}while(ch!=0);

tp->nt=NULL;}

void insertion\_at\_position(){ int ref,i;

printf("\nEnter a position :");

scanf("%d",&ref);

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

printf("Enter data to add:");

scanf("%d",&nn->dt);

tp=st;

for(i=1;i<(ref-1);i++){

tp=tp->nt;

}

nn->nt=tp->nt;

nn->pr=tp;

tp->nt->pr=nn;

tp->nt=nn;

}

void reverse\_display() {

tp = st;

// Move to the end of the list

while (tp != NULL && tp->nt != NULL) {

tp = tp->nt;

}

printf("\nReverse display of elements: ");

while (tp != NULL) {

printf("%d\t", tp->dt);

tp = tp->pr;

}

printf("\n");

}

int main(){

create();

display();

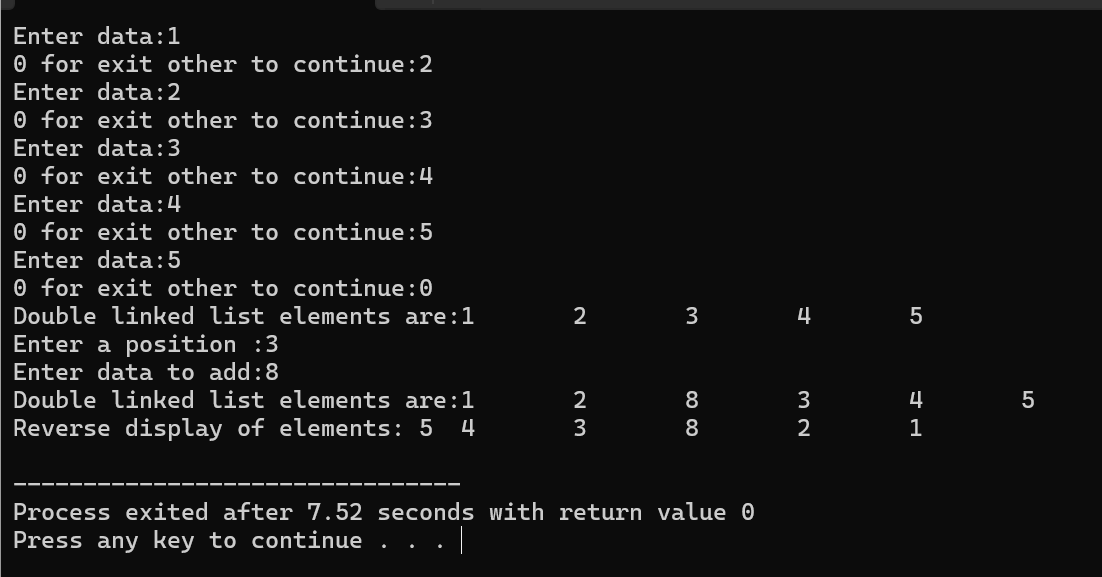
insertion\_at\_position();

display();

reverse\_display();

return 0;}

**OUTPUT:**



**PROBLEM STATEMENT :**

7. Write a C program that uses functions to perform the following:

a. Create a double linked list of integers.

b. Display the list.

c. Delete a node at Beginning and after

**SOURCE CODE:**

/\* 7. Write a C program that uses functions to perform the following:

a. Create a double linked list of integers.

b. Display the list.

c. Delete a node at Beginning and after.\*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next,\*previous;

}\*start=NULL,\*temporary=NULL,\*new\_node=NULL;

void display(){

temporary=start;

printf("\ndouble linked listart elements are:");

while(temporary!=NULL){

printf("%d\t",temporary->data);

temporary=temporary->next;

}

}

void create(){

int choice;

do{

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

printf("Enter data:");

scanf("%d",&new\_node->data);

if(start==NULL){

start=temporary=new\_node;

start->previous=NULL;

}

else {

temporary->next=new\_node;

new\_node->previous=temporary;

temporary=temporary->next;

}

printf("0 for exit other to conextinue:");

scanf("%d",&choice);

}while(choice!=0);

temporary->next=NULL;

}

void delete\_at\_begin(){

temporary=start;

start=temporary->next;

start->previous=NULL;

free(temporary);

}

void delete\_at\_end(){

temporary=start;

while(temporary->next->next!=NULL)

temporary=temporary->next;

new\_node=temporary->next;

temporary->next=NULL;

free(new\_node);

}

void reverse\_display(){

temporary=start;

while(temporary != NULL && temporary->next != NULL)

temporary=temporary->next;

printf("\nReverse display of DLL are: ");

while(temporary!=NULL){

printf("%d ",temporary->data);

temporary=temporary->previous;

}

}

int main(){

create();

display();

delete\_at\_begin();

display();

delete\_at\_end();

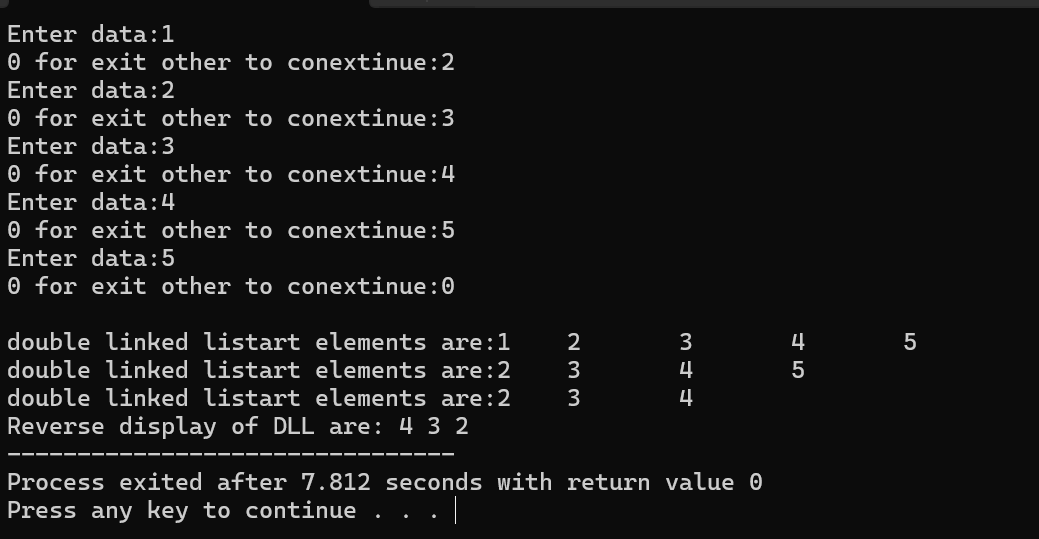
display();

reverse\_display();

return 0;

}

**OUTPUT:**



**PROBLEM STATEMENT :**

8.Write a C program that uses functions to perform the following:

a. Create a double linked list of integers.

b. Display the list.

c. Delete a node at a position

**SOURCE CODE:**

/\* 8.Write a C program that uses functions to perform the following:

a. Create a double linked list of integers.

b. Display the list.

c. Delete a node at a position.\*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next,\*previous;

}\*start=NULL,\*temporary=NULL,\*new\_node=NULL;

void display(){

temporary=start;

printf("\ndouble linked list elements are:");

while(temporary!=NULL){

printf("%d\t",temporary->data);

temporary=temporary->next;

}

}

void create(){

int choice;

do{

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

printf("Enter data:");

scanf("%d",&new\_node->data);

if(start==NULL){

start=temporary=new\_node;

start->previous=NULL;

}

else {

temporary->next=new\_node;

new\_node->previous=temporary;

temporary=temporary->next;

}

printf("0 for exit other to continue:");

scanf("%d",&choice);

}while(choice!=0);

temporary->next=NULL;

}

void delete\_at\_particular\_position(){

int ref,i;

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

scanf("%d",&ref);

temporary=start;

for(i=1;i<(ref-1);i++)

temporary=temporary->next;

new\_node=temporary->next;

new\_node->next->previous=temporary;

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

free(new\_node);

}

void reverse\_display(){

temporary=start;

while(temporary != NULL && temporary->next != NULL)

temporary=temporary->next;

printf("\nReverse display of DLL are: ");

while(temporary!=NULL){

printf("%d ",temporary->data);

temporary=temporary->previous;

}

}

int main(){

create();

display();

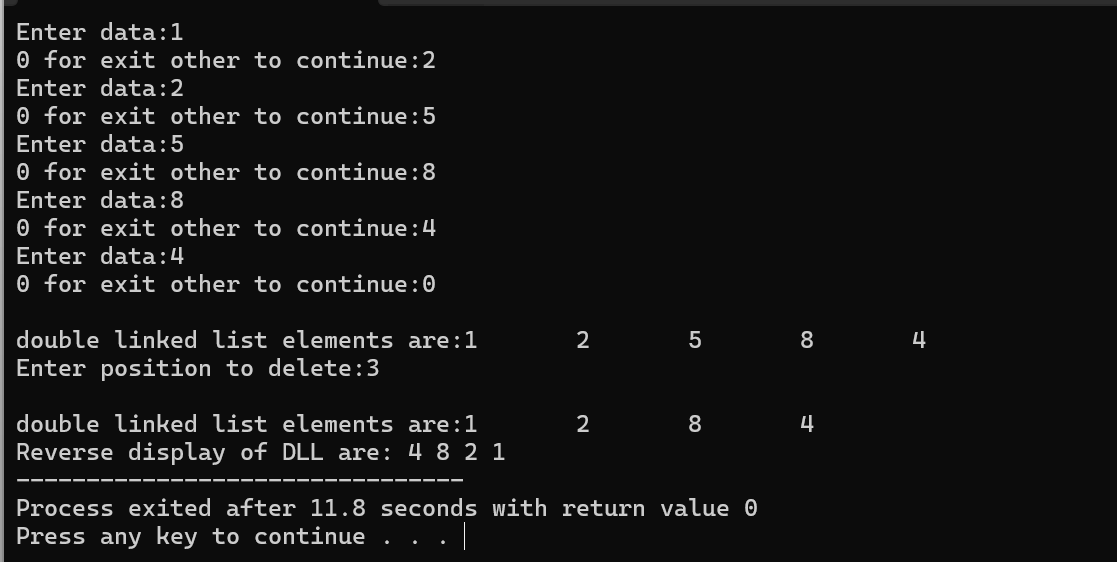
delete\_at\_particular\_position();

display();

reverse\_display();

return 0;

}**OUTPUT:**



**WEEK-5**

**1. Write a C program that uses functions to perform the following:**

**a. Create a Circular single linked list of integers.**

**b. Insert Element in a Circular linked list at begin, Last**

**c. Display the contents of the Circular linked list in reverse.**

/\* a. create CSLL of intergres

b. insert at begin,last

c.display in reverse \*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*nt;

}\*st = NULL,\*tp=NULL,\*nn=NULL;

void Create(){

int ch;

do{

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

printf("Enter element to add: ");

scanf("%d",&nn->dt);

if(st==NULL)

st=tp=nn;

else{

tp->nt=nn;

tp=tp->nt;

}

printf("0 to exit ,other to continue: ");

scanf("%d",&ch);

}while(ch!=0);

tp->nt=st;

}

void Insertion\_at\_Begin(){

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

printf("\nEnter element to insert at begin: ");

scanf("%d",&nn->dt);

tp=st;

do{

tp=tp->nt;

}while(tp->nt!=st);

tp->nt=nn;

nn->nt=st;

st=nn;

}

void Insertion\_at\_End(){

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

printf("\nEnter element to insert at end: ");

scanf("%d",&nn->dt);

tp=st;

while(tp->nt!=st)

tp=tp->nt;

nn->nt=st;

tp->nt=nn;

}

void Display(){

printf("\n CSLL Elements are: ");

tp=st;

do{

printf("%d\t",tp->dt);

tp=tp->nt;

}while(tp!=st);

}

void Display\_Reverse(struct node \*tp){

if(tp->nt==st||tp->nt==NULL)

{printf("\nCSLL Elements are in Reverse Order: %d\t",tp->dt);

return;

}

Display\_Reverse(tp->nt);

printf("%d\t",tp->dt);

}

int main(){

Create();

Display();

Insertion\_at\_Begin();

Insertion\_at\_End();

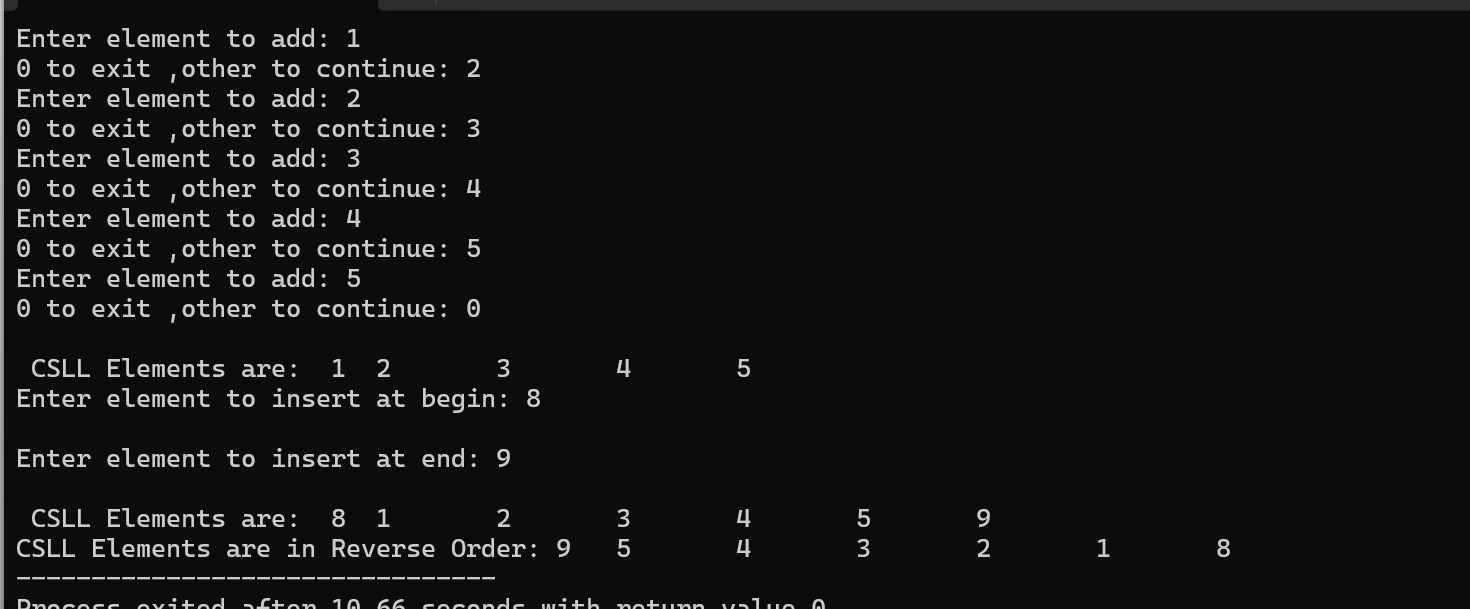
Display();

Display\_Reverse(st);

return 0;

}

**OUTPUT:**



**2. Write a C program that uses functions to perform the following: 1.**

**Create a Circular linked list of characters.**

**2. Display the Nodes**

**3. Delete a node at Beginning and after**

#include<stdio.h>

#include<stdlib.h>

struct node{

char dt;

struct node \*nt;

}\*st = NULL,\*tp=NULL,\*nn=NULL;

void Create(){

int ch;

do{

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

printf("Enter character to add: ");

scanf("%s",&nn->dt);

if(st==NULL)

st=tp=nn;

else{

tp->nt=nn;

tp=tp->nt;

}

printf("0 to exit ,other to continue: ");

scanf("%d",&ch);

}while(ch!=0);

tp->nt=st;

}

void Display(){

printf("\nElements are: ");

tp=st;

do{

printf("%c\t",tp->dt);

tp=tp->nt;

}while(tp!=st);

}

void Delete\_at\_Begin(){

tp=st;

nn=st;

while(tp->nt!=st)

tp=tp->nt;

st=st->nt;

tp->nt=st;

free(nn);

printf("First element is deleted successfully...\n");

}

void Delete\_at\_After\_Element(){

int ref;

printf("Enter after which element to delete: ");

scanf("%s",&ref);

tp=st;

while(tp->dt!=ref)

tp=tp->nt;

nn=tp->nt;

tp->nt=nn->nt;

free(nn);

}

int main(){

Create();

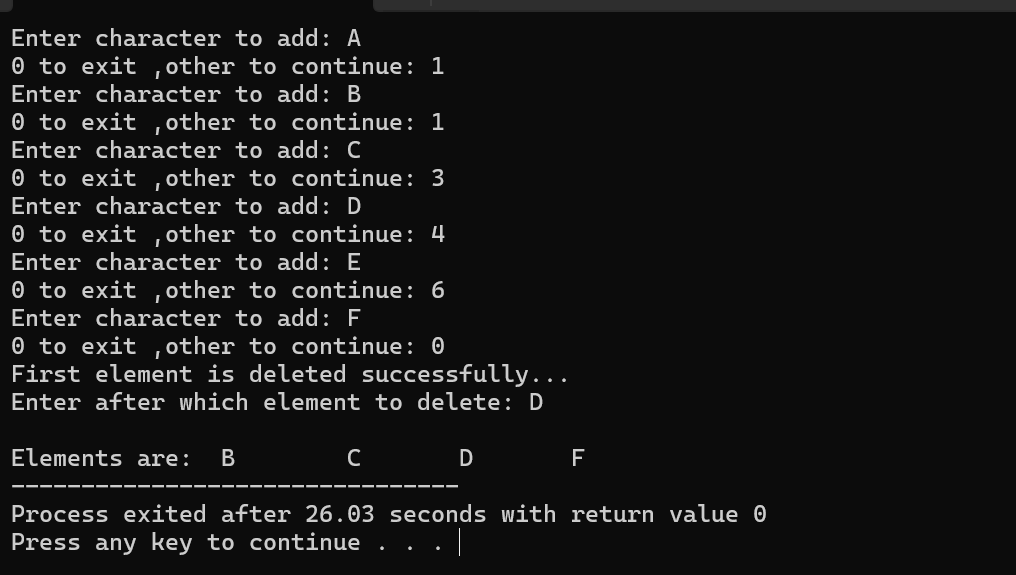
Delete\_at\_Begin();

Delete\_at\_After\_Element();

Display();

return 0;

}**OUTPUT:**



**3.Write a C program that uses functions to perform the following:**

**a. Create a Circular single linked list of integers.**

**b. Insert Element in a Circular linked list at begin and after a node.**

**c. Display the nodes.**

/\*3.Write a C program that uses functions to perform the following:

a. Create a Circular single linked list of integers.

b. Insert Element in a Circular linked list at begin and after a node.

c. Display the nodes.\*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*nt;

}\*st = NULL,\*tp=NULL,\*nn=NULL;

void Create(){

int ch;

do{

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

printf("Enter element to add: ");

scanf("%d",&nn->dt);

if(st==NULL)

st=tp=nn;

else{

tp->nt=nn;

tp=tp->nt;

}

printf("0 to exit ,other to continue: ");

scanf("%d",&ch);

}while(ch!=0);

tp->nt=st;

}

void Display(){

printf("\nElements are: ");

tp=st;

do{

printf("%d\t",tp->dt);

tp=tp->nt;

}while(tp!=st);

}

void Insertion\_at\_Begin(){

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

printf("Enter element to Insert at begin: ");

scanf("%d",&nn->dt);

tp=st;

while(tp->nt!=st)

tp=tp->nt;

tp->nt=nn;

nn->nt=st;

st=nn;

}

void Insertion\_After\_an\_Element(){

int ref;

printf("Enter after which element to insert: ");

scanf("%d",&ref);

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

printf("Enter data to Insert: ");

scanf("%d",&nn->dt);

tp=st;

while(tp->dt!=ref)

tp=tp->nt;

nn->nt=tp->nt;

tp->nt=nn;

}

int main(){

Create();

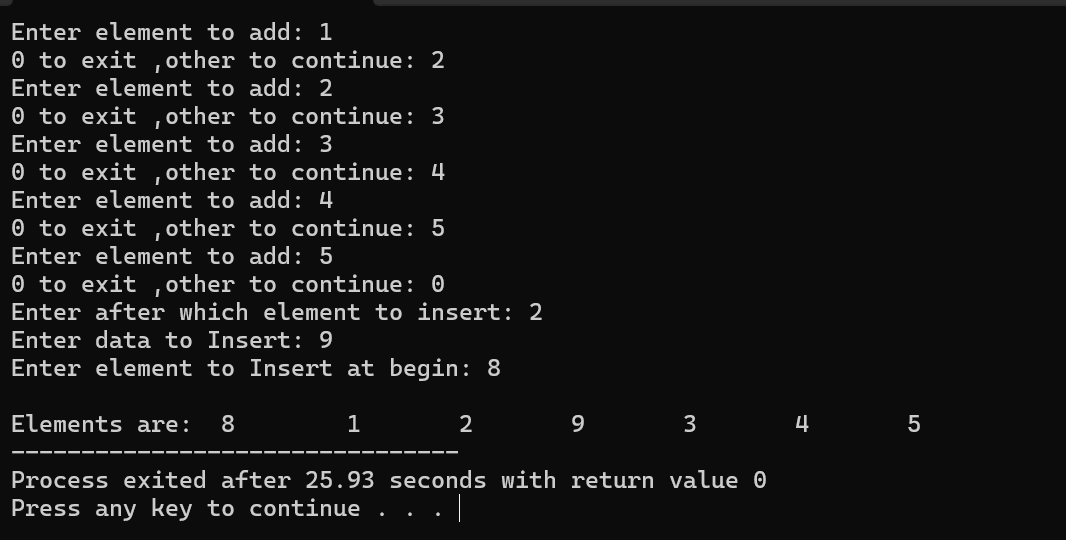
Insertion\_After\_an\_Element();

Insertion\_at\_Begin();

Display();

return 0;

}



**4.Write a C program that uses functions to perform the following:**

**a. Create a Circular Single linked list of integers.**

**b. Display the Nodes.**

**c. Delete a node at Beginning and after**

/\* 4.Write a C program that uses functions to perform the following:

a. Create a Circular Single linked list of integers.

b. Display the Nodes.

c. Delete a node at Beginning and after

\*/

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*nt;

}\*st = NULL,\*tp=NULL,\*nn=NULL;

void Create(){

int ch;

do{

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

printf("Enter element to add: ");

scanf("%d",&nn->dt);

if(st==NULL)

st=tp=nn;

else{

tp->nt=nn;

tp=tp->nt;

}

printf("0 to exit ,other to continue: ");

scanf("%d",&ch);

}while(ch!=0);

tp->nt=st;

}

void Display(){

printf("\nElements are: ");

tp=st;

do{

printf("%d\t",tp->dt);

tp=tp->nt;

}while(tp!=st);

}

void Delete\_begin(){

nn=st;

st=st->nt;

tp->nt=st;

free(nn);

}

void Delete\_After\_Given\_Element(){

int ref;

printf("\nEnter after which element to delete: ");

scanf("%d",&ref);

tp=st;

while(tp->dt!=ref)

tp=tp->nt;

nn=tp->nt;

tp->nt=nn->nt;

free(nn);

}

int main(){

Create();

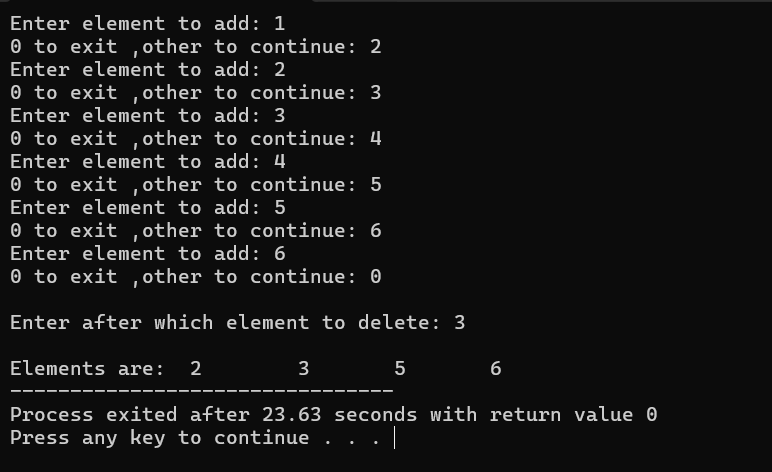
Delete\_begin();

Delete\_After\_Given\_Element();

Display();

return 0;

}**OUTPUT:**

****

**5.Write a C program that uses functions to perform the following:**

**a. Create a Double linked list of integers.**

**b. Delete an element at a position.**

**c. Display the contents of the above list.**

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*nt,\*pr;

}\*st = NULL,\*tp=NULL,\*nn=NULL;

void Create(){

int ch;

do{

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

printf("Enter element to add: ");

scanf("%d",&nn->dt);

if(st==NULL){

st=tp=nn;

st->pr=NULL;

}

else{

tp->nt=nn;

nn->pr=tp;

tp=tp->nt;

}

printf("0 to exit ,other to continue: ");

scanf("%d",&ch);

}while(ch!=0);

tp->nt=NULL;

}

void Display(){

printf("\nElements are: ");

tp=st;

do{

printf("%d\t",tp->dt);

tp=tp->nt;

}while(tp!=NULL);

}

void Delete\_At\_ParticularPosition(){

int ref,i;

printf("\nEnter particular position to delete: ");

scanf("%d",&ref);

tp=st;

for(i=1;i<(ref-1);i++)

tp=tp->nt;

nn=tp->nt;

tp->nt=nn->nt;

nn->nt->pr=tp;

free(nn);

}

int main(){

Create();

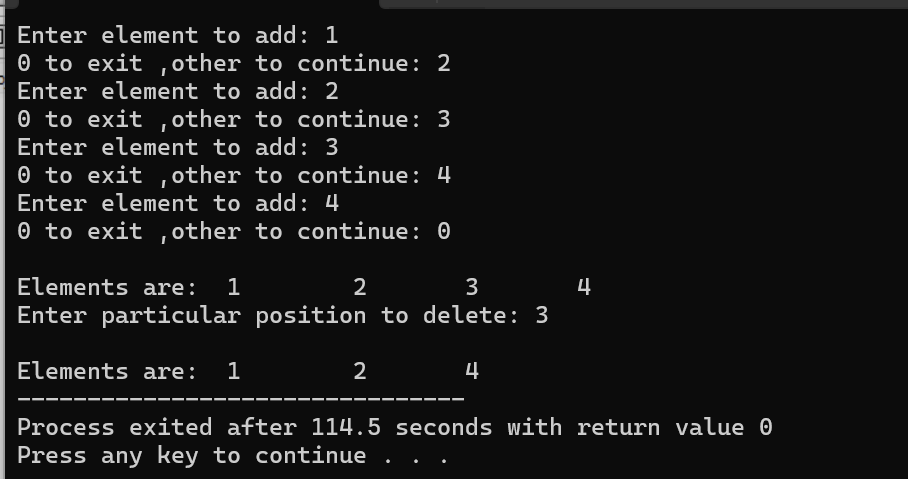
Display();

Delete\_At\_ParticularPosition();

Display();

}

**OUTPUT:**



**WEEK -6:**

**PROBLEM STATEMENT :**

1.Write a C program that uses recursive function to perform insertion sort

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#define n 5

int a[n]={5,9,7,8,17}, i=1, j, temp;

void InsSort()

{

temp=a[i];

j=i-1;

while(j>=0 && a[j]>temp)

{

a[j+1]=a[j];

j--;

}

a[j+1]=temp;

i++;

if(i<n)

InsSort();

}

int main()

{

InsSort();

printf("The SORTED LIST: ");

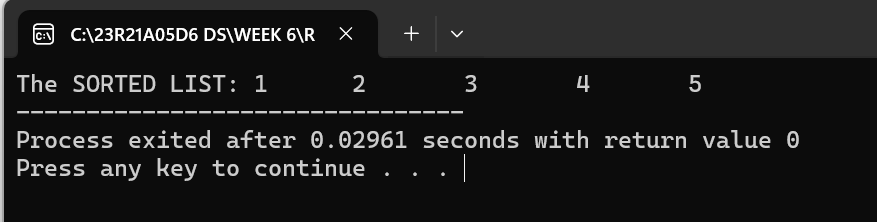
for(i=0;i<n;i++)

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

return 0;

}

**OUTPUT:**



**PROBLEM STATEMENT :**

2.Write a C program that uses non-recursive function to perform optimized bubble sort

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#define n 5

int a[n]={1,4,5,8,7},count=0,i,j,temp,flag;

void bubble()

{

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

{

count++;

flag=0;

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

if(a[j]>a[j+1])

{

flag=1;

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

if(flag==0)

break;

}

}

int main()

{

bubble();

printf("Number of passes: %d\n",count);

printf("The bubble sorted array is: ");

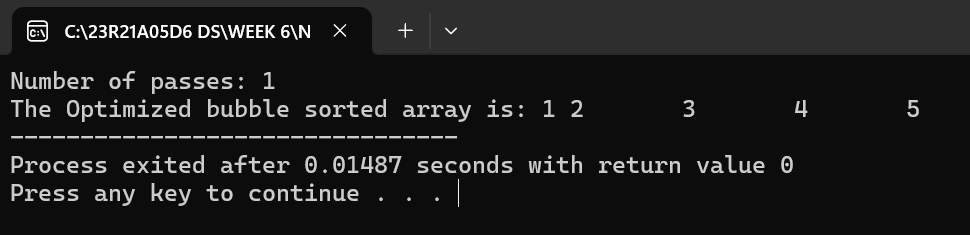
for(i=0;i<n;i++)

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

return 0;

}

**OUTPUT:**

****

**PROBLEM STATEMENT :**

3.Write a C program that uses non-recursive function to perform selection sort.

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#define n 5

int a[n]={12,19,7,1,17}, i, j, min, temp;

void select()

{

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

{

min=i;

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

{

if(a[j]<a[min])

min=j;}

if(min!=i)

{

temp=a[i];

a[i]=a[min];

a[min]=temp;

}

}

}

int main()

{

select();

printf("The result of selection sort is: ");

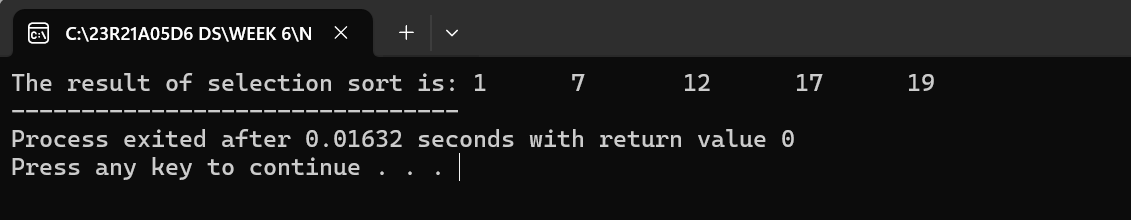
for(i=0;i<n;i++)

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

return 0;

}

**OUTPUT:**

****

**PROBLEM STATEMENT:**

**. Write a C program that uses non-recursive function to perform insertion sort**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#define n 5

int a[n]={5,9,7,8,17}, i, j, temp;

void InsSort()

{

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

temp=a[i];

j=i-1;

while(j>=0 && a[j]>temp)

{

a[j+1]=a[j];

j--;

}

a[j+1]=temp;

}

}

int main()

{

InsSort();

printf("The SORTED LIST: ");

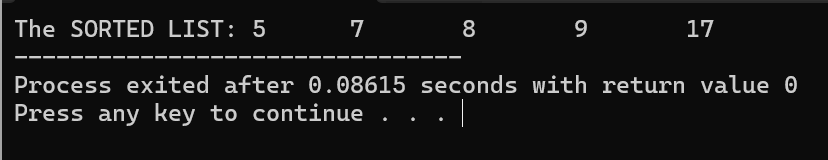
for(i=0;i<n;i++)

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

return 0;

}

**OUTPUT:**



**PROBLEM STATEMENT:**

**. Write a C program that uses recursive function to perform selection sort**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#define n 5

int a[n]={12,19,7,1,17}, i=0, j, min, temp;

void select()

{

min=i;

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

{

if(a[j]<a[min])

min=j;}

if(min!=i)

{

temp=a[i];

a[i]=a[min];

a[min]=temp;

}

i++;

if(i<n)

select();

}

int main()

{

select();

printf("The result of selection sort is: ");

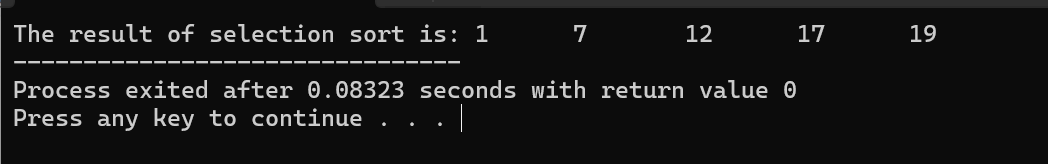
for(i=0;i<n;i++)

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

return 0;

}

**OUTPUT:**



**PROBLEM STATEMENT:**

**. Write a C program that uses recursive function to perform optimized bubble sort**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

#define n 5

int a[n]={1,2,3,4,5},count=0,i=0,j,temp,flag;

void bubble()

{

count++;

flag=0;

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

if(a[j]>a[j+1])

{

flag=1;

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

i++;

if(flag==0)

return;

if(i<n-1)

bubble();

}

int main()

{

bubble();

printf("Number of passes: %d\n",count);

printf("The bubble sorted array is: ");

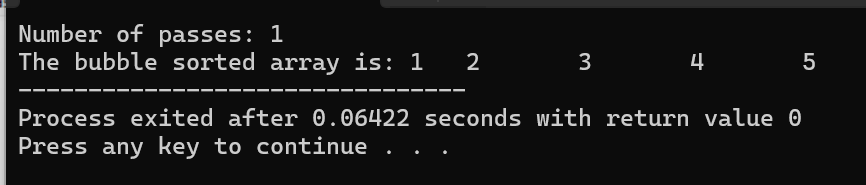
for(i=0;i<n;i++)

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

return 0;

}

**OUTPUT:**



**WEEK -7:**

**PROBLEM STATEMENT**:

1.write a C program to implement Quick sort.

**SOURCE CODE:**

#include<stdio.h>

#define N 5

int a[N]={5,4,3,2,1},l,r,temp,key,h=0,t=N-1,i;

void Quick\_Sort(int h,int t){

if(h<t){

l=h;

r=t;

key=h;}

while(l<r){

while(a[l]<=a[key] && l<t)

l++;

while(a[r]>a[key] && r>h)

r--;

if(l<r){

temp=a[l];

a[l]=a[r];

a[r]=temp;

}

temp=a[r];

a[r]=a[key];

a[key]=temp;

Quick\_Sort(h,r-1);

Quick\_Sort(r+1,t);}}

int main(){

Quick\_Sort(0,N-1);

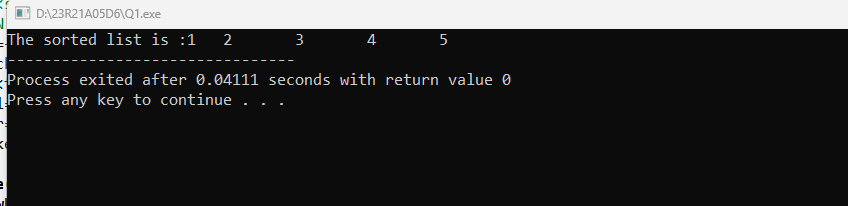
printf("The sorted list is :");

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

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

return 0;}

**OUTPUT:**



**PROBLEM STATEMENT:**

2) Write a C program to implement Merge sort

**SOURCE CODE:**

#include<stdio.h>

#define N 10

int a[N]={9,8,7,6,5,4,3,2,1,10},k,b[N],i,j,l,r,mid,temp;

void MergeSort(int l,int mid,int r){

i=l;

j=mid+1;

k=l;

while(i<=mid && j<=r){

if(a[i]<=a[j]){

b[k]=a[i];

i++;}

else{

b[k]=a[j];

j++;}

k++;}

while(i<=mid){

b[k]=a[i];

i++;

k++;}

while(j<=r){

b[k]=a[j];

j++;

k++;}

for(i=l;i<k;i++)

a[i]=b[i];}

void Sort(int l ,int r){

int mid;

if(l<r){ mid=(l+r)/2;

Sort(l,mid);

Sort(mid+1,r);

MergeSort(l,mid,r);}}

int main(){

Sort(0,N-1);

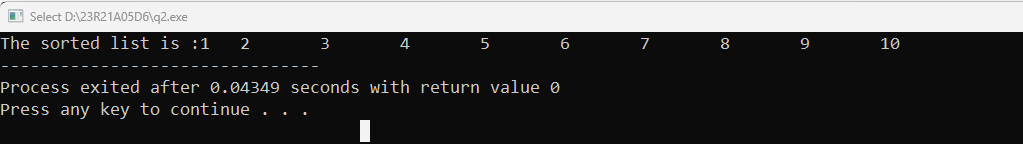
printf("The sorted list is :");

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

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

return 0;}

**OUTPUT:**



**PROBLEM STATEMENT:**

2) Write a C program to implement Heap sort

**SOURCE CODE:**

#include<stdio.h>

#define N 10

int a[N]={9,8,7,6,5,4,3,2,1,10},i,j,max,temp,l,r;

void Heap(){

for(j=N;j>1;j--){

for(i=(j/2)-1;i>=0;i--){

l=2\*i+1;

r=2\*i+2;

max=i;

if(a[l]>a[max]&& l<j){

max=l;

}

if(a[r]>a[max]&& r<j)

max=r;

if(max!=i){

temp=a[max];

a[max]=a[i];

a[i]=temp;

}

}

temp=a[0];

a[0]=a[j-1];

a[j-1]=temp;

}}

int main(){

Heap();

printf("The sorted list is :");

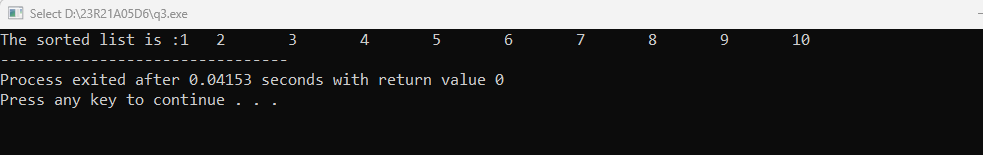
for(i=0;i<N;i++)

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

return 0;

}

**OUTPUT:**

****

**WEEK 8:**

**PROBLEM STATEMENT :**

Develop a C program USING LINKED LIST to perform the following operations using Binary Tree.

Insertion of an element into Binary Tree.

Display all the elements in **Preorder traversal** from Binary Tree.

**Note: The values/ elements should be Integer values**

**SOURCE CODE:**

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

int dt;

struct node \*l,\*r;};

struct node \* New\_Node(int data){

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

nn->dt=data;

nn->l=NULL;

nn->r=NULL;

return nn;}

void Display\_Preorder(struct node \*tp){

if(tp!=NULL){

printf("%d ",tp->dt);

Display\_Preorder(tp->l);

Display\_Preorder(tp->r);

}

}

int main(){

struct node \*root=New\_Node(1);

root->l=New\_Node(2);

root->r=New\_Node(3);

root->l->l=New\_Node(4);

root->l->r=New\_Node(5);

root->r->l=New\_Node(6);

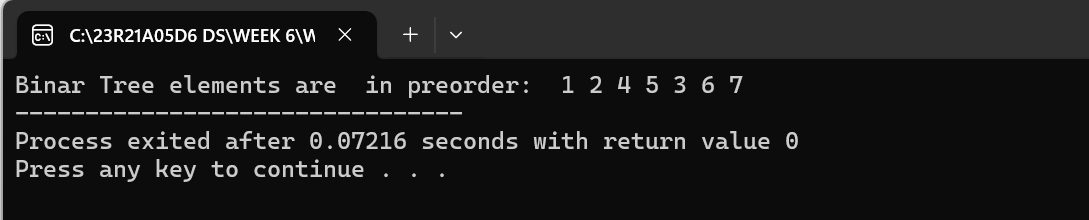
root->r->r=New\_Node(7);

printf("Binar Tree elements are in preorder : ");

Display\_Preorder(root);

return 0;

}**OUTPUT:**



**PROBLEM STATEMENT :**

Develop a C program USING LINKED LIST to perform the following operations using Binary Tree.

Insertion of an element into Binary Tree.

Display all the elements in **Inorder traversal** from Binary Tree.

**Note: The values/ elements should be Character values**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*l,\*r;

};

struct node \* New\_Node(int data){

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

nn->dt=data;

nn->l=NULL;

nn->r=NULL;

return nn;

}

void Display\_Inorder(struct node \*tp){ if(tp!=NULL){

Display\_Inorder(tp->l);

printf("%d ",tp->dt);

Display\_Inorder(tp->r);

}

}

int main(){

struct node \*root=New\_Node(1);

root->l=New\_Node(2);

root->r=New\_Node(3);

root->l->l=New\_Node(4);

root->l->r=New\_Node(5);

root->r->l=New\_Node(6);

root->r->r=New\_Node(7);

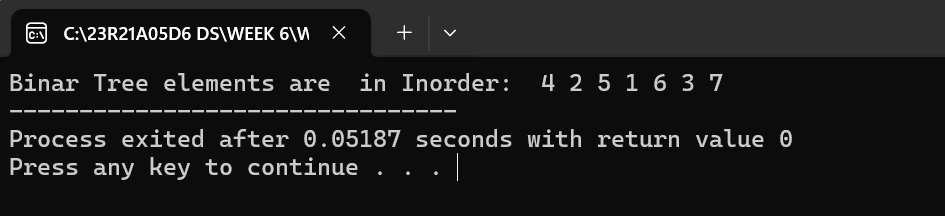
printf("Binar Tree elements are in Inorder: ");

Display\_Inorder(root);

return 0;

}

**OUTPUT:**



**PROBLEM STATEMENT :**

Develop a C program USING LINKED LIST to perform the following operations using Binary Tree.

Insertion of an element into Binary Tree.

Display all the elements in **Postorder traversal** from Binary Tree.

**Note: The values/ elements should be character values**

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*l,\*r;

};

struct node \* New\_Node(int data){

struct node \*nn = (struct node \*)malloc(sizeof(struct node)); nn->dt=data;

nn->l=NULL; nn->r=NULL;

return nn;

}

void Display\_Postorder(struct node \*tp){

if(tp!=NULL){

Display\_Postorder(tp->l);

Display\_Postorder(tp->r);

printf("%d ",tp->dt);

}

}

int main(){

struct node \*root=New\_Node(1);

root->l=New\_Node(2);

root->r=New\_Node(3);

root->l->l=New\_Node(4);

root->l->r=New\_Node(5);

root->r->l=New\_Node(6);

root->r->r=New\_Node(7);

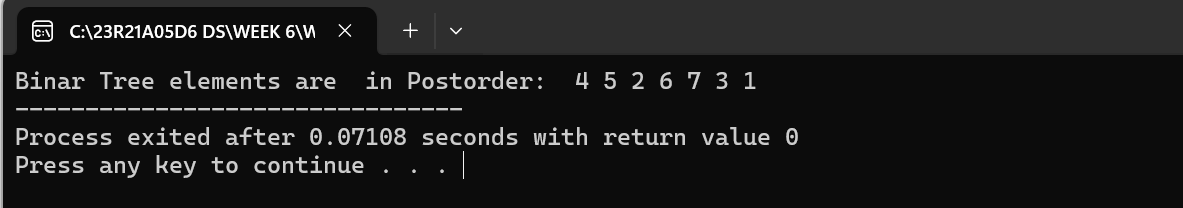
printf("Binar Tree elements are in Postorder: ");

Display\_Postorder(root);

return 0;

}

**OUTPUT:**



**WEEK 9:**

**PROBLEM STATEMENT :**

1.Develop a C program USING LINKED LIST to perform the following operations in Binary Search Tree. 1. Create a Binary Search Tree. 2. Search for a given element. Note: The values/ elements should be Integer values

**SOURCE CODE:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int dt;

struct node \*l,\*r;

}\*root=NULL,\*tp=NULL,\*nn=NULL;

void Insert(){

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

printf("Enter value: ");

scanf("%d",&nn->dt);

nn->l=NULL;

nn->r=NULL;

if(root==NULL)

root=nn;

else{

tp=root;

while(tp->l!=nn && tp->r!=nn)

if(nn->dt < tp->dt)

if(tp->l!=NULL)

tp=tp->l;

else

tp->l=nn;

else

if(tp->r!=NULL)

tp=tp->r;

else

tp->r=nn;

}

}

void Display(struct node \*tp){

if(tp!=NULL){

Display(tp->l);

printf("%d\t",tp->dt);

Display(tp->r);

}

}

void search(){

int s;

printf("\nEnter element: ");

scanf("%d", &s);

tp=nn=root;

while(s!=nn->dt){ tp=nn;

nn=(s<nn->dt)? tp->l: tp->r;

if(nn==NULL){printf("Element not found\n");

return;}}

printf("Element found\n");}

int main(){

int ch;

do{

Insert();

printf("0 to exit other to continue: ");

scanf("%d",&ch);

}while(ch!=0);

printf("Binary Search Tree elements: ");

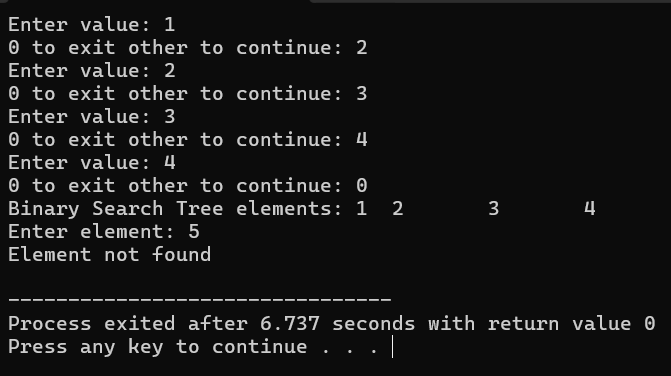
Display(root);

search();

return 0;

}

**OUTPUT:**

****

**PROBLEM STATEMENT:**

2.Develop a C program USING LINKED LIST to perform the following operations using Binary Search Tree. 1. Deletion of an element in Binary Search Tree. 2. Display all the elements in order. Note: The values/ elements should be Integer values

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

struct node{

int dt;

struct node \*l, \*r;

}\*root=NULL, \*tp=NULL, \* nn=NULL;

void insert(){ nn=malloc(sizeof(struct node));

printf("Enter val: ");

scanf("%d", &nn->dt);

nn->l = nn->r = NULL;

if(root==NULL)

root=nn;

else{ tp=root;

while(tp->l!=nn && tp->r!=nn)

if(nn->dt<tp->dt)

if(tp->l!=NULL)

tp=tp->l;

else tp->l=nn;

else if(tp->r!=NULL) tp=tp->r;

else tp->r=nn;}}

void search(){

int s;

printf("Enter element: ");

scanf("%d", &s);

tp=nn=root;

while(s!=nn->dt){ tp=nn;

nn=(s<nn->dt)? tp->l: tp->r;

if(nn==NULL){printf("Element not found\n");

return;}}

printf("Element found\n");}

void del(){

search();

if(nn==NULL){ printf("Enter valid input, ");

search();}

if(nn->r==NULL){ //Node with 0 child or 1 left child

if(nn==root) root=root->l;

else{ if(tp->l==nn)

tp->l=nn->l;

else tp->r=nn->l;}

free(nn);}

else if(nn->l==NULL){ //Node with 1 right child

if(nn==root) root=root->r;

else{ if(tp->l==nn)

tp->l=nn->r;

else tp->r=nn->r;}

free(nn);}

else{ //Node with 2 child nodes

tp=nn->r;

while(tp->l!=NULL)

tp=tp->l;

nn->dt=tp->dt;

if(nn->r!=tp){

nn=nn->r;

while(nn->l!=tp)

nn=nn->l;

nn->l=tp->r;}

else nn->r=tp->r;

free(tp);}}

void disp(struct node\* tp){

if(tp!=NULL){

disp(tp->l);

printf("%d ", tp->dt);

disp(tp->r);}}

int main(){ int ch;

do{ printf("Enter 0 to exit, 1 to insert, 2 to search,3 to delete, 4 to disp: ");

scanf("%d", &ch);

switch(ch){ case 1:insert(); break;

case 2:printf("To search, ");

search();

break;

case 3:printf("To delete, ");

del();

printf("and deleted\n ");

break;

case 4:printf("Tree elements using inorder traversal: ");

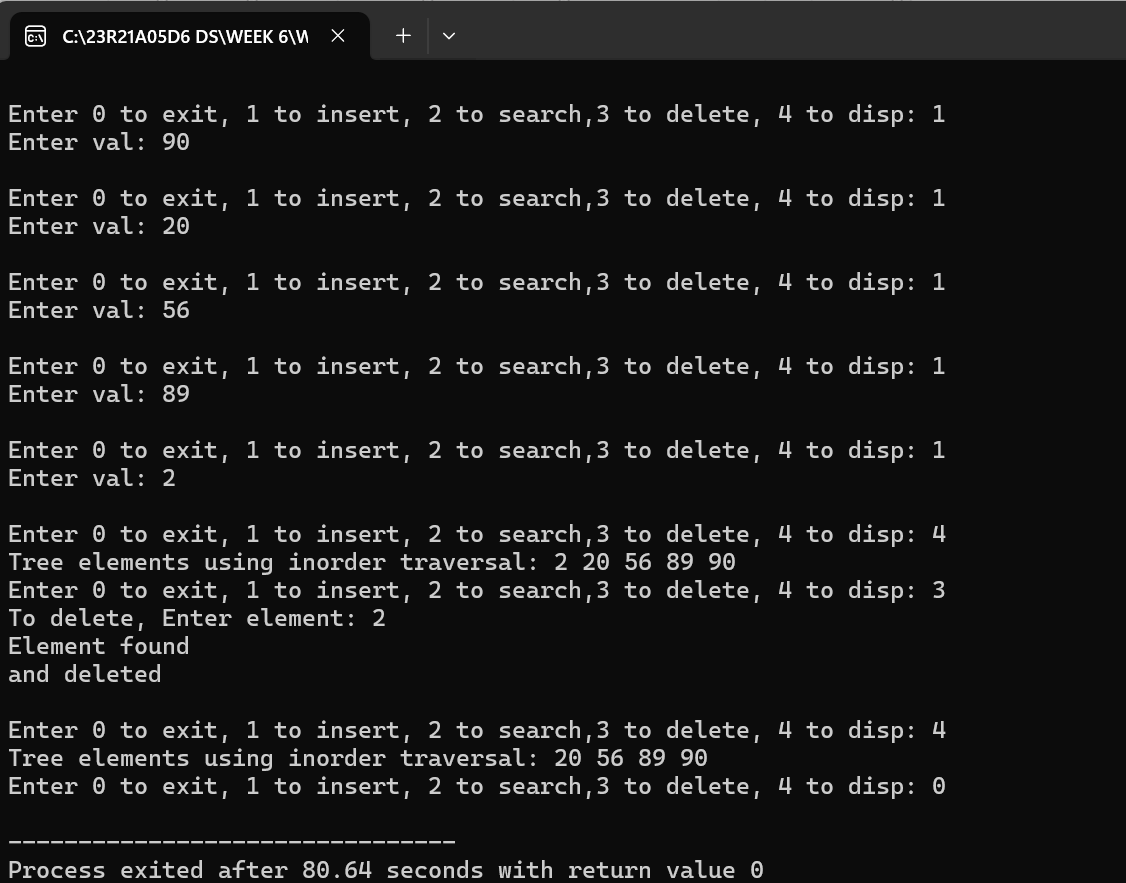
disp(root);break;

case 0: break;

}}while(ch!=0);

return 0;}

**OUTPUT:**



**PROBLEM STATEMENT:**

3.Develop a C program USING LINKED LIST to perform the following operations using Binary Search Tree. 1. Deletion of an element in Binary Search Tree. 2. Display all the elements greater than the deleted element. Note: The values/ elements should be Integer values

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

int s;

struct node{

int dt;

struct node \*l, \*r;

}\*root=NULL,\*tp=NULL,\*nn=NULL,\*dn=NULL;

void insert(){

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

printf("Enter val:");

scanf("%d",&nn->dt);

nn->l=NULL;

nn->r=NULL;

if(root==NULL)

root=nn;

else{

tp=root;

while((nn->dt<tp->dt && tp->l!=NULL) || (nn->dt>tp->dt && tp->r!=NULL)){

if(nn->dt<tp->dt && tp->l!=NULL)

tp=tp->l;

if(nn->dt>tp->dt && tp->r!=NULL)

tp=tp->r;

}

if(nn->dt<tp->dt)

tp->l=nn;

else

tp->r=nn;

}}

struct node\* search(struct node \*tp){

printf("Enter element: ");

scanf("%d",&s);

while(tp!=NULL){

if(tp->dt==s){

printf("Element available ");

return tp;

}

else if(s<tp->dt)

tp=tp->l;

else

tp=tp->r;

}

printf("Element not available");

return NULL;

}

void del(){

dn=search(root);

if(dn!=NULL){

if(dn->l!=NULL && dn->r!=NULL){

tp=dn->l;

nn=tp;

while(nn->r!=NULL){

tp=nn;

nn=tp->r;

}

if(tp!=nn)

tp->r=nn->l;

else

dn->l=nn->l;

dn->dt=nn->dt;

free(nn);

}

else{

if(dn==root){

if(root->l!=NULL)

root=root->l;

else

root=root->r;

}

else{

tp=root;

while(tp->l!=dn && tp->r!=dn){

if(s<tp->dt)

tp=tp->l;

else

tp=tp->r;

}

if (tp->l==dn){

if(dn->r!=NULL)

tp->l=dn->r;

else

tp->l=dn->l;

}

else{

if(dn->r!=NULL)

tp->r=dn->r;

else

tp->r=dn->l;

}}

free(dn);

dn=NULL;

}}

printf(" To Delete");}

void disp(struct node \*tp){

if(tp!=NULL){

disp(tp->l);

if(tp->dt>s) // To Display all the elements greater than the deleted element

printf("%d ", tp->dt);

disp(tp->r);

}}

int main() {

int ch;

do{

printf("\nEnter 0 to exit, 1 to insert, 2 to disp, 3 to search, 4 to del : ");

scanf("%d",&ch);

switch(ch){

case 0: break;

case 1: insert();

break;

case 2: disp(root);

break;

case 3: search(root);

break;

case 4: del();

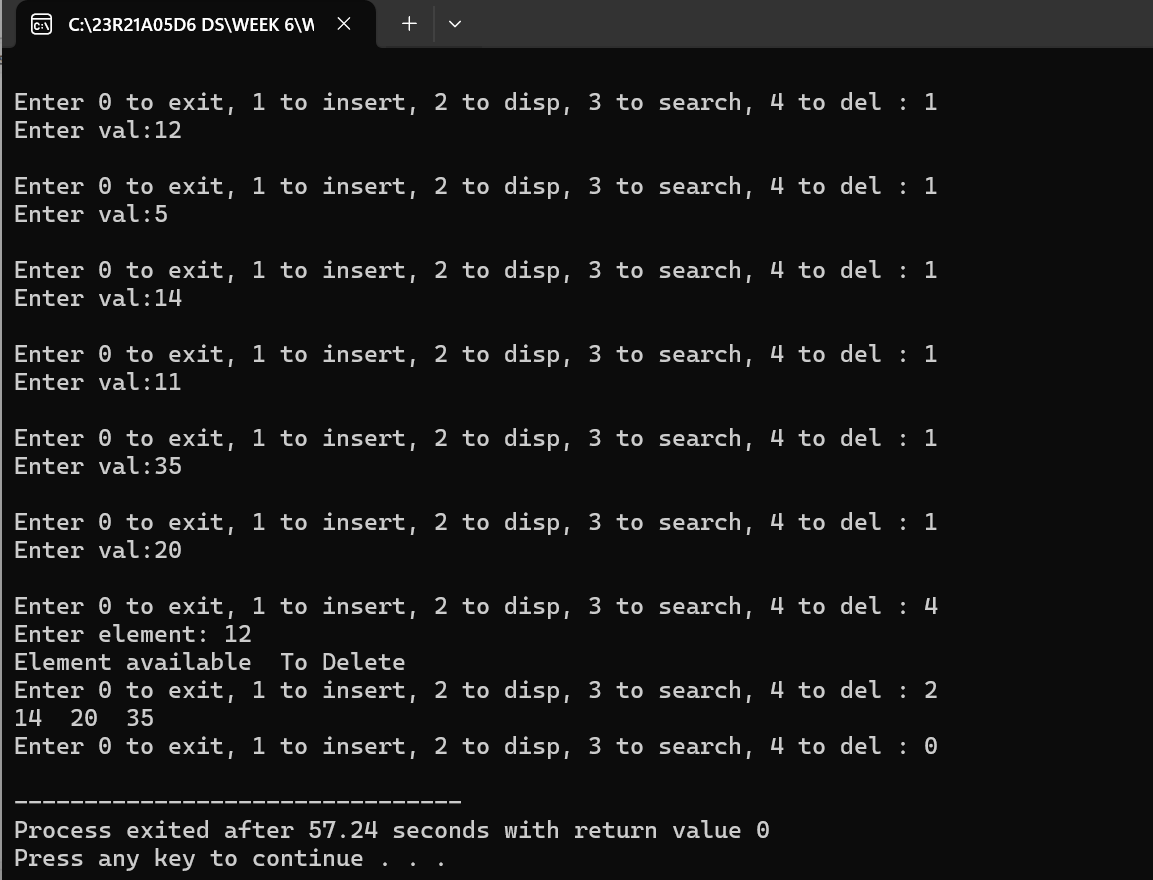
break;

}}while(ch!=0);

return 0;

}

**OUTPUT:**



**WEEK 10:**

**PROBLEM STATEMENT :**

1.Develop a C program USING LINKED LIST to perform the following operations in AVL Tree. 1. Create an AVL Tree. 2. Display all the elements in order. Note: The values/ elements should be Integer values

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

// Define a structure for a node in the AVL tree

struct Node {

int value; // Value of the node

struct Node\* left; // Pointer to the left child

struct Node\* right; // Pointer to the right child

int height; // Height of the node (used for balancing)

};

// Function to get the height of a node

int height(struct Node\* node) {

if (node == NULL)

return 0;

return node->height;

}

// Function to get the maximum of two integers

int max(int a, int b) {

return (a > b) ? a : b;

}

// Function to create a new node with the given value

struct Node\* newNode(int value) {

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

node->value = value;

node->left = node->right = NULL;

node->height = 1; // New node is initially at height 1

return node;

}

// Right rotate utility to balance the tree

struct Node\* rightRotate(struct Node\* y) {

struct Node\* x = y->left;

struct Node\* T2 = x->right;

// Perform rotation

x->right = y;

y->left = T2;

// Update heights

y->height = max(height(y->left), height(y->right)) + 1;

x->height = max(height(x->left), height(x->right)) + 1;

// Return the new root

return x;

}

// Left rotate utility to balance the tree

struct Node\* leftRotate(struct Node\* x) {

struct Node\* y = x->right;

struct Node\* T2 = y->left;

// Perform rotation

y->left = x;

x->right = T2;

// Update heights

x->height = max(height(x->left), height(x->right)) + 1;

y->height = max(height(y->left), height(y->right)) + 1;

// Return the new root

return y;

}

// Get balance factor of node N

int getBalance(struct Node\* N) {

if (N == NULL)

return 0;

return height(N->left) - height(N->right);

}

// Insert a new value into the AVL tree

struct Node\* insert(struct Node\* node, int value) {

// 1. Perform normal BST insert

if (node == NULL)

return(newNode(value));

if (value < node->value)

node->left = insert(node->left, value);

else if (value > node->value)

node->right = insert(node->right, value);

else

return node; // Duplicate values are not allowed in AVL tree

// 2. Update the height of the ancestor node

node->height = 1 + max(height(node->left), height(node->right));

// 3. Get the balance factor of this ancestor node

int balance = getBalance(node);

// If the node becomes unbalanced, there are 4 cases:

// Left Left Case

if (balance > 1 && value < node->left->value)

return rightRotate(node);

// Right Right Case

if (balance < -1 && value > node->right->value)

return leftRotate(node);

// Left Right Case

if (balance > 1 && value > node->left->value) {

node->left = leftRotate(node->left);

return rightRotate(node);

}

// Right Left Case

if (balance < -1 && value < node->right->value) {

node->right = rightRotate(node->right);

return leftRotate(node);

}

// return the (unchanged) node pointer

return node;

}

void InOrder(struct Node\* root) {

if (root != NULL) {InOrder(root->left);

printf("%d ", root->value);

InOrder(root->right);

}

}

// Driver program to test above functions

int main() {

struct Node\* root = NULL;

int choice;

do{ printf("1.Insert\n2.Display in in order\n0.Exit\nenter your choice: ");

scanf("%d",&choice);

switch (choice) {

case 1:

int value;

printf("Enter data: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf("In-order traversal of the AVL tree:\n");

InOrder(root);

printf("\n");

break;

case 0:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice! Please try again.\n");

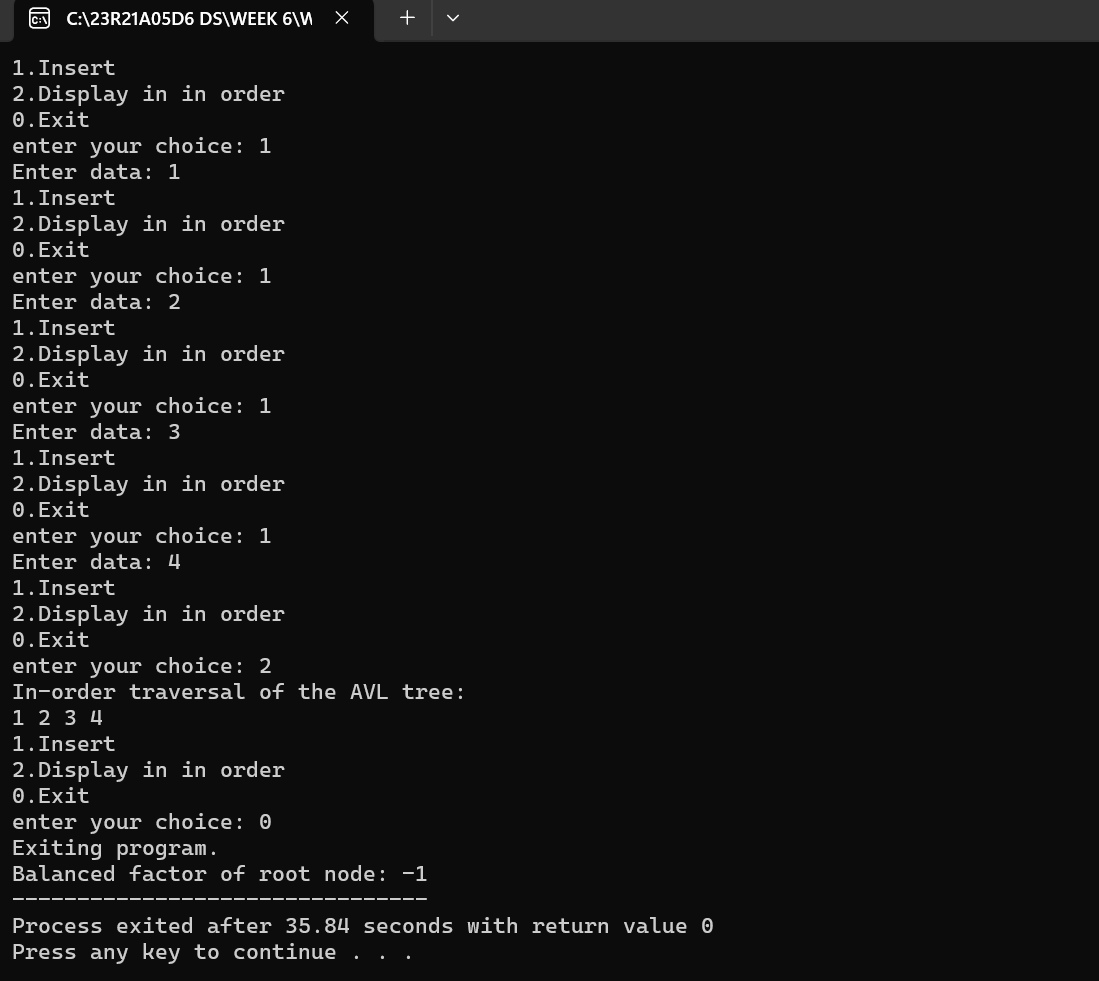
}

} while (choice != 0);

printf("Balanced factor of root node: %d",getBalance(root));

return 0;}

**OUTPUT:**



**PROBLEM STATEMENT :**

2. Develop a C program USING LINKED LIST to perform the following operations in AVL Tree. 1. Create an AVL Tree. 2. Search for a given element. Note: The values/ elements should be Integer values

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

// Define a structure for a node in the AVL tree

struct Node {

int value; // Value of the node

struct Node\* left; // Pointer to the left child

struct Node\* right; // Pointer to the right child

int height; // Height of the node (used for balancing)

};

// Function to get the height of a node

int height(struct Node\* node) {

if (node == NULL)

return 0;

return node->height;

}

// Function to get the maximum of two integers

int max(int a, int b) {

return (a > b) ? a : b;

}

// Function to create a new node with the given value

struct Node\* newNode(int value) {

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

node->value = value;

node->left = node->right = NULL;

node->height = 1; // New node is initially at height 1

return node;

}

// Right rotate utility to balance the tree

struct Node\* rightRotate(struct Node\* y) {

struct Node\* x = y->left;

struct Node\* T2 = x->right;

// Perform rotation

x->right = y;

y->left = T2;

// Update heights

y->height = max(height(y->left), height(y->right)) + 1;

x->height = max(height(x->left), height(x->right)) + 1;

// Return the new root

return x;

}

// Left rotate utility to balance the tree

struct Node\* leftRotate(struct Node\* x) {

struct Node\* y = x->right;

struct Node\* T2 = y->left;

// Perform rotation

y->left = x;

x->right = T2;

// Update heights

x->height = max(height(x->left), height(x->right)) + 1;

y->height = max(height(y->left), height(y->right)) + 1;

// Return the new root

return y;

}

// Get balance factor of node N

int getBalance(struct Node\* N) {

if (N == NULL)

return 0;

return height(N->left) - height(N->right);

}

// Insert a new value into the AVL tree

struct Node\* insert(struct Node\* node, int value) {

// 1. Perform normal BST insert

if (node == NULL)

return(newNode(value));

if (value < node->value)

node->left = insert(node->left, value);

else if (value > node->value)

node->right = insert(node->right, value);

else

return node; // Duplicate values are not allowed in AVL tree

// 2. Update the height of the ancestor node

node->height = 1 + max(height(node->left), height(node->right));

// 3. Get the balance factor of this ancestor node

int balance = getBalance(node);

// If the node becomes unbalanced, there are 4 cases:

// Left Left Case

if (balance > 1 && value < node->left->value)

return rightRotate(node);

// Right Right Case

if (balance < -1 && value > node->right->value)

return leftRotate(node);

// Left Right Case

if (balance > 1 && value > node->left->value) {

node->left = leftRotate(node->left);

return rightRotate(node);

}

// Right Left Case

if (balance < -1 && value < node->right->value) {

node->right = rightRotate(node->right);

return leftRotate(node);

}

// return the (unchanged) node pointer

return node;

}

void InOrder(struct Node\* root) {

if (root != NULL) {

InOrder(root->left);

printf("%d ", root->value);

InOrder(root->right);

}

}

// Function to search for a value in the AVL tree

struct Node\* search(struct Node\* root, int value) {

// Base case: root is NULL or the value is found

if (root == NULL || root->value == value)

return root;

// If the value is smaller than root's value, search in the left subtree

if (value < root->value)

return search(root->left, value);

// Otherwise, search in the right subtree

return search(root->right, value);

}

// Driver program to test above functions

int main() {

struct Node\* root = NULL;

int choice, value;

do {

printf("1. Insert\n2. Display in-order\n3. Search\n0. Exit\nEnter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data: ");

scanf("%d", &value);

root = insert(root, value);

break;

case 2:

printf("In-order traversal of the AVL tree:\n");

InOrder(root);

printf("\n");

break;

case 3:

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

scanf("%d", &value);

struct Node\* result = search(root, value);

if (result != NULL)

printf("Value %d found in the AVL tree.\n", value);

else

printf("Value %d not found in the AVL tree.\n", value);

break;

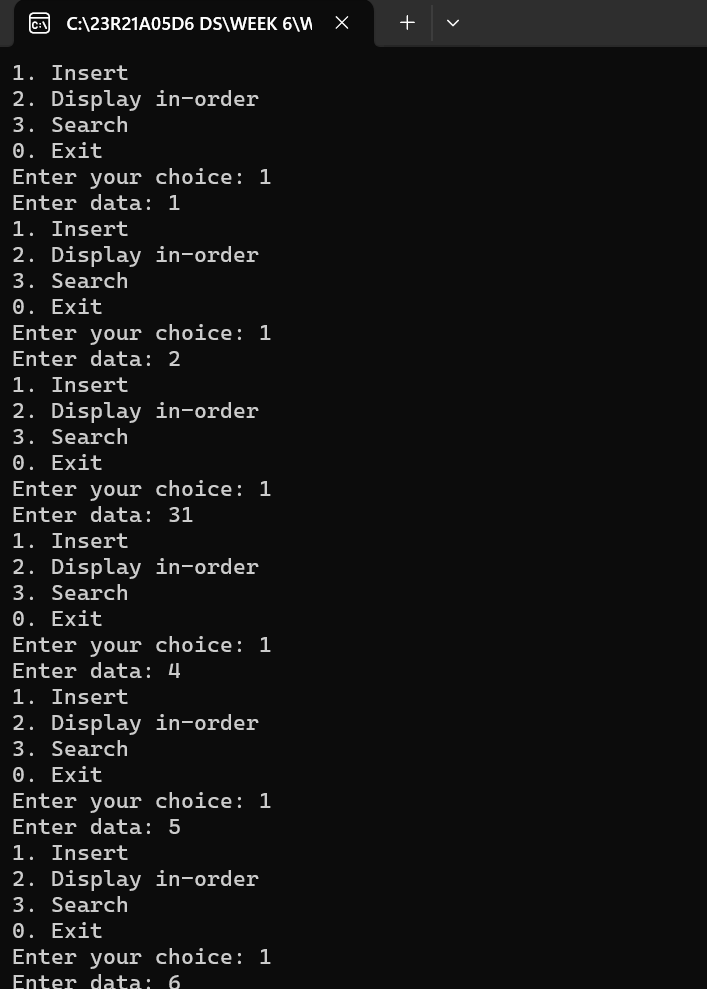
}

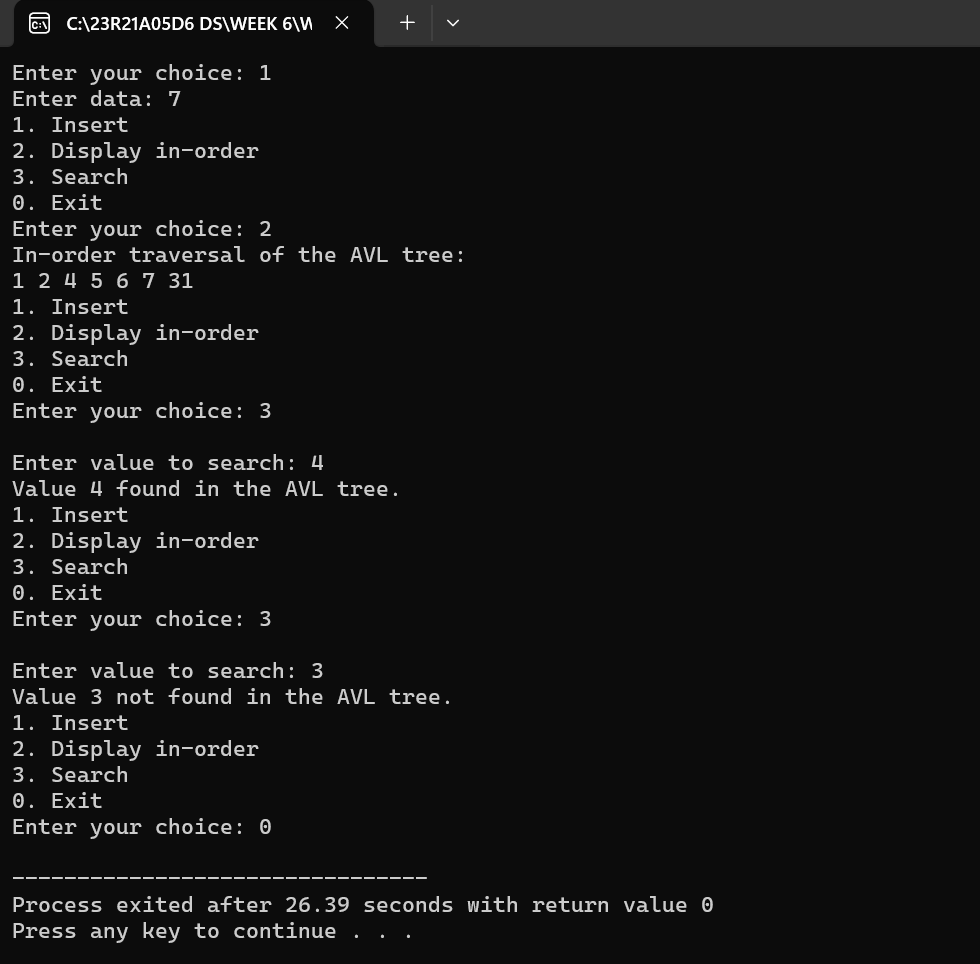
} while (choice != 0);

return 0;

}

**OUTPUT:**





**PROBLEM STATEMENT :**

3.Develop a C program USING LINKED LIST to perform the following operations using AVL Tree. 1. Deletion of an element in AVL Tree. 2. Display all the elements in order. Note: The values/ elements should be Integer values

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

struct node {

int value;

struct node\* left;

struct node\* right;

int height;

};

int height(struct node\* tp) {

if (tp == NULL)

return 0;

return tp->height;

}

int max(int a, int b) {

return (a > b) ? a : b;

}

int getBalance(struct node\* tp) {

if (tp == NULL)

return 0;

return height(tp->left) - height(tp->right);

}

struct node\* createNode(int value) {

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

tp->value = value;

tp->left = NULL;

tp->right = NULL;

tp->height = 1;

return tp;

}

struct node\* rightRotate(struct node\* tp) {

struct node\* x = tp->left;

tp->left = x->right;

x->right = tp;

tp->height = max(height(tp->left), height(tp->right)) + 1;

x->height = max(height(x->left), height(x->right)) + 1;

return x;

}

struct node\* leftRotate(struct node\* tp) {

struct node\* y = tp->right;

tp->right = y->left;

y->left = tp;

tp->height = max(height(tp->left), height(tp->right)) + 1;

y->height = max(height(y->left), height(y->right)) + 1;

return y;

}

struct node\* insert(struct node\* tp, int value) {

if (tp == NULL)

return createNode(value);

if (value < tp->value)

tp->left = insert(tp->left, value);

else if (value > tp->value)

tp->right = insert(tp->right, value);

else

return tp;

tp->height = 1 + max(height(tp->left), height(tp->right));

int balance = getBalance(tp);

if (balance > 1 && value < tp->left->value)

return rightRotate(tp);

if (balance > 1 && value > tp->left->value) {

tp->left = leftRotate(tp->left);

return rightRotate(tp);

}

if (balance < -1 && value > tp->right->value)

return leftRotate(tp);

if (balance < -1 && value < tp->right->value) {

tp->right = rightRotate(tp->right);

return leftRotate(tp);

}

return tp;

}

struct node\* deleteNode(struct node\* root, int value) {

if (root == NULL)

return root;

if (value < root->value)

root->left = deleteNode(root->left, value);

else if (value > root->value)

root->right = deleteNode(root->right, value);

else {

if (root->left == NULL)

return root->right;

else if (root->right == NULL)

return root->left;

struct node\* temp = root->right;

while (temp->left != NULL) {

temp = temp->left;

}

root->value = temp->value;

root->right = deleteNode(root->right, temp->value);

}

if (root == NULL)

return root;

root->height = 1 + max(height(root->left), height(root->right));

int balance = getBalance(root);

if (balance > 1 && getBalance(root->left) >= 0)

return rightRotate(root);

if (balance > 1 && getBalance(root->left) < 0) {

root->left = leftRotate(root->left);

return rightRotate(root);

}

if (balance < -1 && getBalance(root->right) <= 0)

return leftRotate(root);

if (balance < -1 && getBalance(root->right) > 0) {

root->right = rightRotate(root->right);

return leftRotate(root);

}

return root;

}

void inOrder(struct node\* root) {

if (root != NULL) {

inOrder(root->left);

printf("%d ", root->value);

inOrder(root->right);

}

}

int main() {

struct node\* root = NULL;

int choice, value;

do {

printf("\n1. Insert\n2. Delete\n3. Display (In-order)\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

root = insert(root, value);

printf("%d inserted into the AVL tree.\n", value);

break;

case 2:

printf("Enter value to delete: ");

scanf("%d", &value);

root = deleteNode(root, value);

printf("%d deleted from the AVL tree.\n", value);

break;

case 3:

printf("In-order traversal of the AVL tree: ");

inOrder(root);

printf("\n");

break;

case 4: printf("Tree height is %d\n Tree root value is %d",root->height, root->value);

printf("Exiting program.\n");

exit(0);

default:

printf("Invalid choice! Please try again.\n");

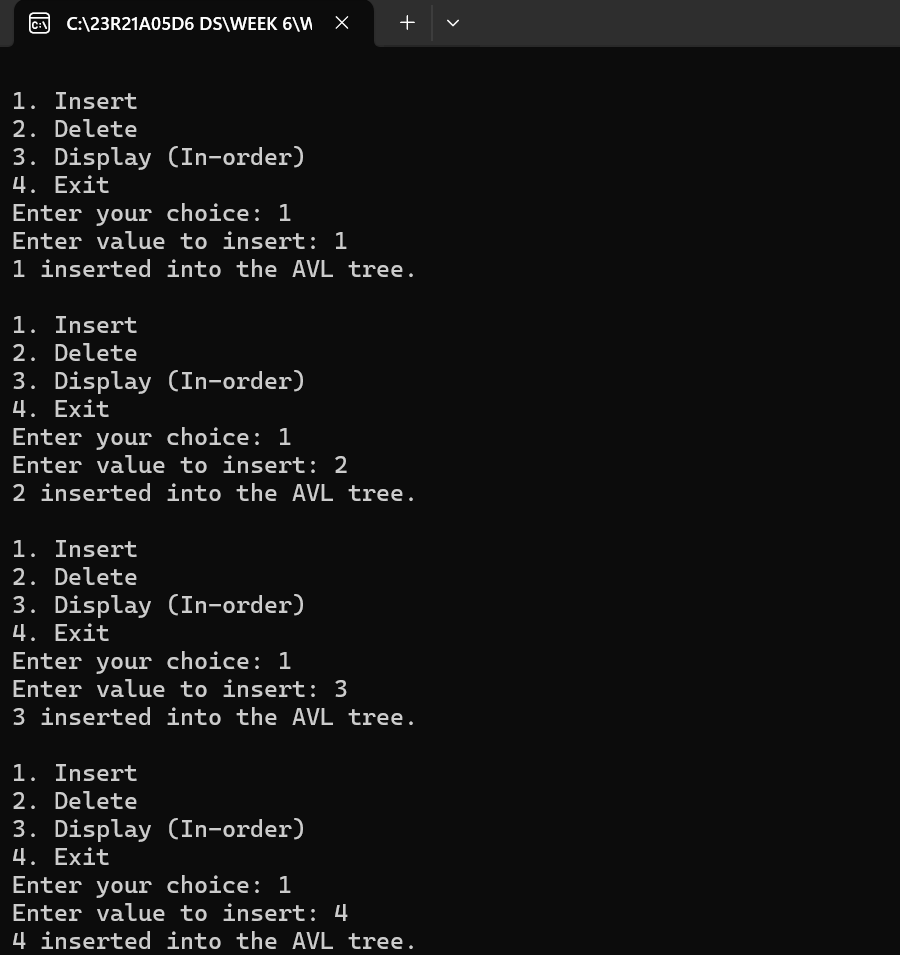
}

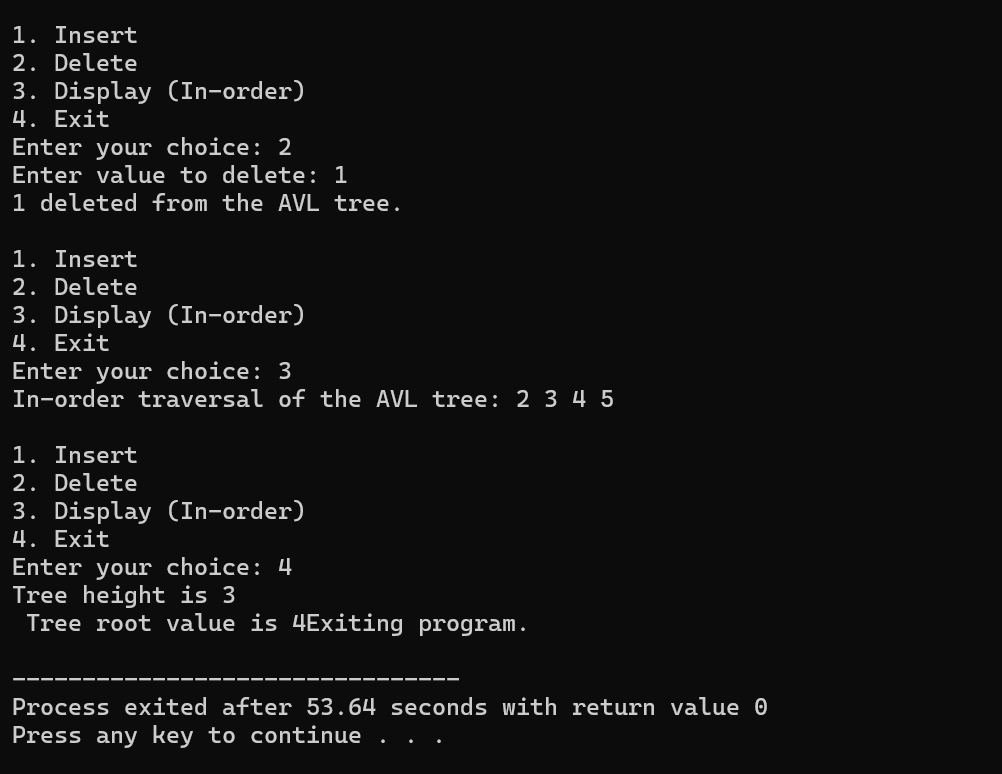
} while(choice!=0);

return 0;

}

**OUTPUT:**





**WEEK -11**

**PROBLEM STATEMENT:**

**Develop a C program to implement the BFS graph traversal.**

**SOURCE CODE:**

#include <stdio.h>

#define MAX 10

int queue[MAX],rr=-1,fr=-1,adj[MAX][MAX],visited[MAX]={0},i,j,n,start;

int main() {

printf("Enter the number of vertices:");

scanf("%d", &n);

printf("Enter the adjacency matrix: \n");

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

for (j=0;j<n;j++)

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

printf("Enter starting node:");

scanf("%s", &start);

start=start-65;

queue[++rr]=start;

++fr;

visited[start]=1;

printf("BFS traversal:");

while (rr>=fr) {

start=queue[fr++];

printf("%c ",start+65);

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

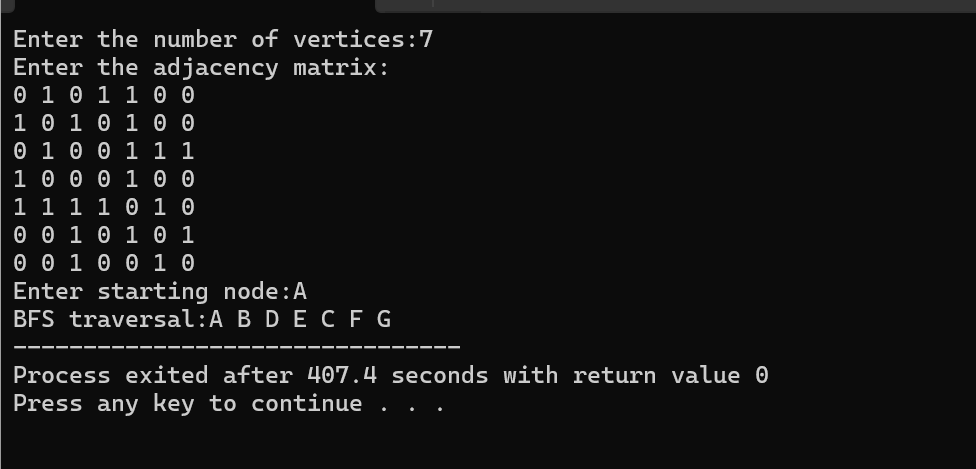
if(adj[start][i]&&visited[i]==0){

queue[++rr]=i;

visited[i]=1;

}}}}

**OUTPUT:**



**PROBLEM STATEMENT:**

**Develop a C program to implement the DFS graph traversal.**

**SOURCE CODE:**

#include <stdio.h>

#define MAX 10

int stack[MAX],top = -1,adj[MAX][MAX],visited[MAX]={0}, i,j,n,start;

int main() {

printf("Enter number of vertices:");

scanf("%d",&n);

printf("Enter the adjacency matrix: \n");

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

for(j=0;j<n;j++)

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

printf("enter starting node: ");

scanf("%s",&start);

start=start-65;

top++;

stack[top]=start;

visited[start]=1;

printf("DFS Traversal: ");

printf("%c ",start+65);

while(top!=-1){

start=stack[top];

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

if(adj[start][i]==1&&visited[i]==0){

top++;

stack[top]=i;

visited[i]=1;

printf("%c ",i+65);

break;

}}

if(i==n)

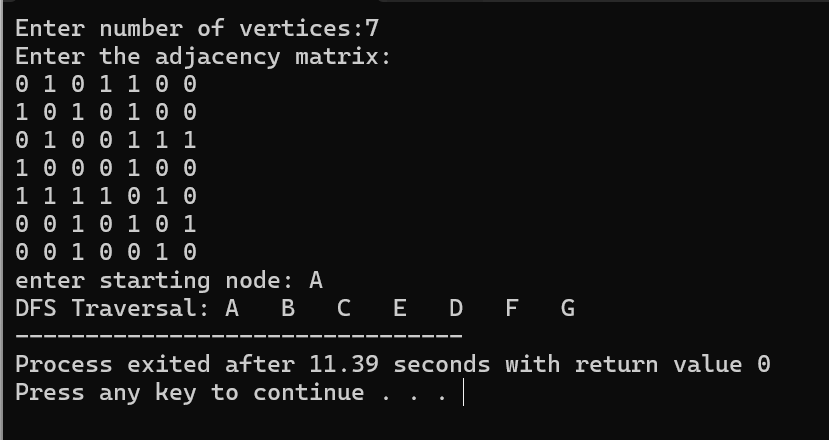
top--;

}

return 0;

}

**OUTPUT:**



**WEEK -12**

**PROBLEM STATEMENT:**

**1.Develop a C program to implement Pattern matching using Brute force.**

**SOURCE CODE:**

#include <stdio.h>

#include <string.h>

int main()

{

int i,j;

char text[] = "Michaelic";

char patt[] = "ic";

int M = strlen(patt), N = strlen(text);

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

for (j = 0; j < M; j++)

if (text[i + j] != patt[j])

break;

if (j==M)

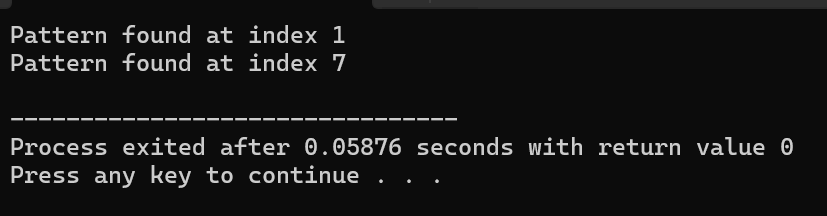
printf("Pattern found at index %d \n", i);

}

return 0;

}

**OUTPUT:**



**PROBLEM STATEMENT:**

**2.Develop a C program to implement Knuth-Morris-Pratt Pattern matching.**

**SOURCE CODE:**

#include <stdio.h>

#include <string.h>

int main() {

char text[] = "ababdababdabcdcabfabc";

char pattern[] = "abcdcabfabc";

// Length of the pattern and the text

int LP = strlen(pattern);

int LS = strlen(text);

// Array to hold the LPS (Longest Prefix Suffix) values

int lps[LP];

// Step 1: Build the LPS array

int length = 0;

int i = 1;

lps[0] = 0; // LPS for the first character is always 0

while (i < LP) {

if (pattern[i] == pattern[length]) {

length++;

lps[i] = length;

i++;

} else {

if (length != 0) {

length = lps[length - 1];

} else {

lps[i] = 0;

i++;

}

}

}

i = 0;

int j = 0;

printf("The pattern is found at the index:\n");

while (i < LS) {

if (pattern[j] == text[i]) {

j++;

i++;

}

if (j == LP) {

printf("Pattern found at index %d\n", i - j);

j = lps[j - 1];

} else if (i < LS && pattern[j] != text[i]) {

if (j != 0) {

j = lps[j - 1];

} else {

// If j is 0, move to the next character in the text

i = i + 1;

}

}

}

printf("\nThe following is the LPS Table:\n");

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

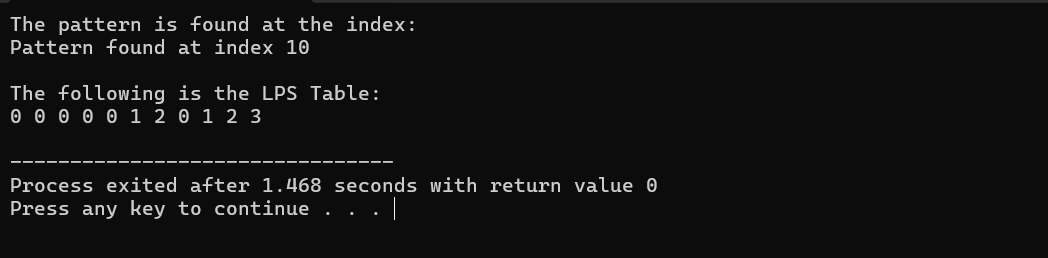
printf("%d ", lps[i]);

printf("\n");

return 0;

}

**OUTPUT:**

****