TASK 3

Data structures and Algorithm

Name:Sparsh Arya
Registeration Number: 17BEC0656
Teacher:Gayatri P

Question 1
Menu driven C program to implement stack using linked list

Pseudo Code:

1. PUSH:

```
void push ()
{
   int val;
   struct node *ptr =(struct node*)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("not able to push the element");
   }
   else
   {
      printf("Enter the value");
      scanf("%d",&val);
      if(head==NULL)
      {
         ptr->val = val;
         ptr -> next = NULL;
         head=ptr;
      }
      else
      {
         ptr->val = val;
         ptr->next = head;
         head=ptr;

      }
      printf("Item pushed");

      }
   }
```

2. POP:

```
void pop()
{
   int item;
   struct node *ptr;
   if (head == NULL)
   {
      printf("Underflow");
   }
   else
   {
```

```
        item = head->val;
        ptr = head;
        head = head->next;
        free(ptr);
        printf("Item popped");

    } }
```

3. TRAVERSING:

```
void display()
{
    int i;
    struct node *ptr;
    ptr=head;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}
```

C Code:

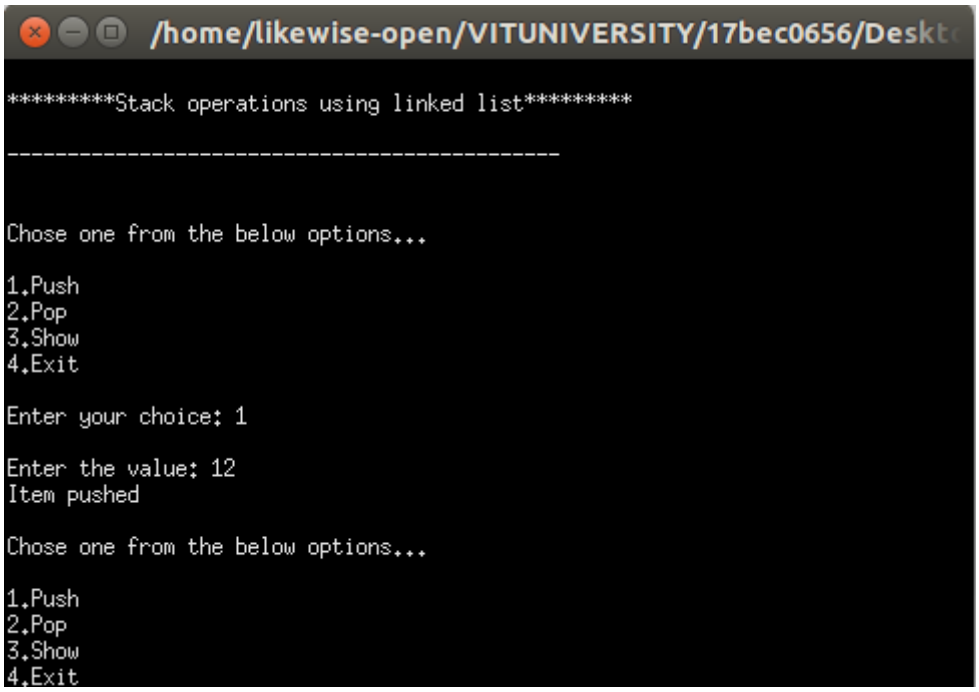
```
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
{
int val;
struct node *next;
};
struct node *head;

void main ()
{
    int choice=0;
    printf("\n*********Stack operations using linked list*********\n");
```

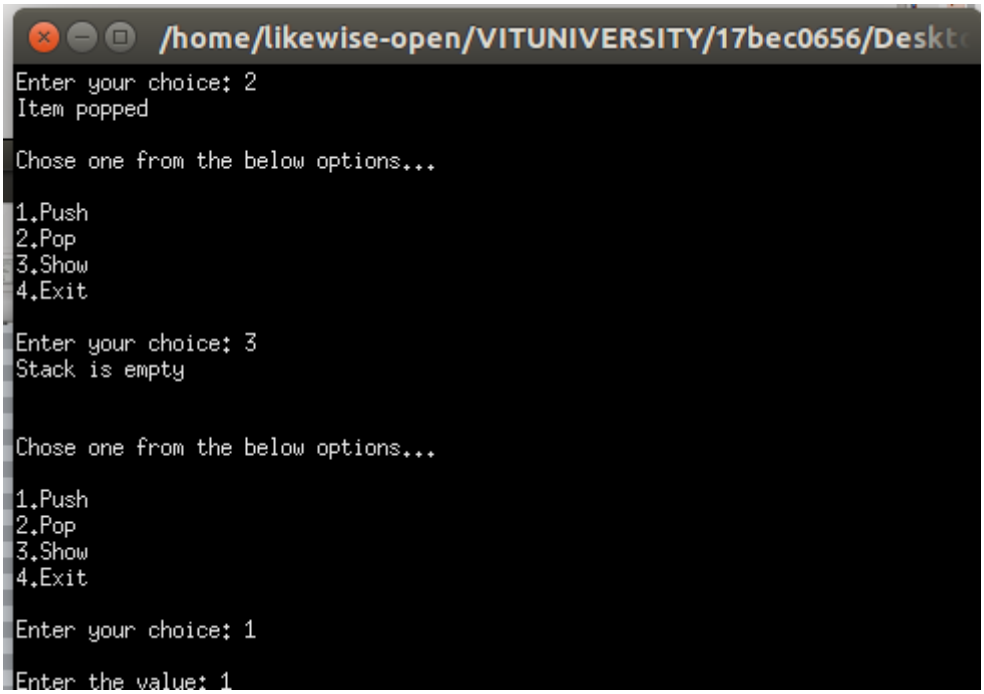
```
    printf("\n---------------------------------------------\n");
    while(choice != 4)
    {
        printf("\n\nChose one from the below options...\n");
        printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
        printf("\n Enter your choice \n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            case 4:
            {
                printf("Exiting....");
                break;
            }
            default:
            {
                printf("Please Enter valid choice ");
            }
    };
}
}
void push ()
{
    int val;
    struct node *ptr = (struct node*)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("not able to push the element");
    }
    else
    {
        printf("Enter the value");
        scanf("%d",&val);
        if(head==NULL)
```

```
      {
         ptr->val = val;
         ptr -> next = NULL;
         head=ptr;
      }
      else
      {
         ptr->val = val;
         ptr->next = head;
         head=ptr;

      }
      printf("Item pushed");

   }
}

void pop()
{
   int item;
   struct node *ptr;
   if (head == NULL)
   {
      printf("Underflow");
   }
   else
   {
      item = head->val;
      ptr = head;
      head = head->next;
      free(ptr);
      printf("Item popped");

   }
}
void display()
{
   int i;
   struct node *ptr;
   ptr=head;
   if(ptr == NULL)
   {
      printf("Stack is empty\n");
   }
   else
   {
      printf("Printing Stack elements \n");
      while(ptr!=NULL)
      {
```

```
            printf("%d\n",ptr->val);
            ptr = ptr->next;
        }
    }
}
```

Output:





Question 2
Menu driven C program to implement queue using linked list.

Pseudo Code:
1. Create an empty queue

```
create()
{
   front = rear = NULL;
}
```

2. Returns queue size

```
queuesize()
{
   printf("\n Queue size : %d", count);
}
```

3. Enqueing the queue

```
enq(int data)
{
   if (rear == NULL)
   {
      rear = (struct node *)malloc(1*sizeof(struct node));
      rear->ptr = NULL;
      rear->info = data;
      front = rear;
   }
   else
   {
      temp=(struct node *)malloc(1*sizeof(struct node));
      rear->ptr = temp;
      temp->info = data;
      temp->ptr = NULL;

      rear = temp;
   }
   count++;
}
```

4. Displaying the queue elements

```
void display()
{
   front1 = front;

   if ((front1 == NULL) && (rear == NULL))
   {
      printf("Queue is empty");
      return;
   }
```

```
   while (front1 != rear)
   {
      printf("%d ", front1->info);
      front1 = front1->ptr;
   }
   if (front1 == rear)
      printf("%d", front1->info);
}
```

5. Dequeing the queue

```
deq()
{
   front1 = front;

   if (front1 == NULL)
   {
      printf("\n Error: Trying to display elements from empty queue");
      return;
   }
   else
      if (front1->ptr != NULL)
      {
         front1 = front1->ptr;
         printf("\n Dequed value : %d", front->info);
         free(front);
         front = front1;
      }
      else
      {
         printf("\n Dequed value : %d", front->info);
         free(front);
         front = NULL;
         rear = NULL;
      }
      count--;
}
```

6. Returns the front element of queue

```
frontelement()
{
   if ((front != NULL) && (rear != NULL))
      return(front->info);
   else
      return 0;
}
```

7. Display if queue is empty or not

```
empty()
{
    if ((front == NULL) && (rear == NULL))
       printf("\n Queue empty");
   else
      printf("Queue not empty");
}
```

C Code:

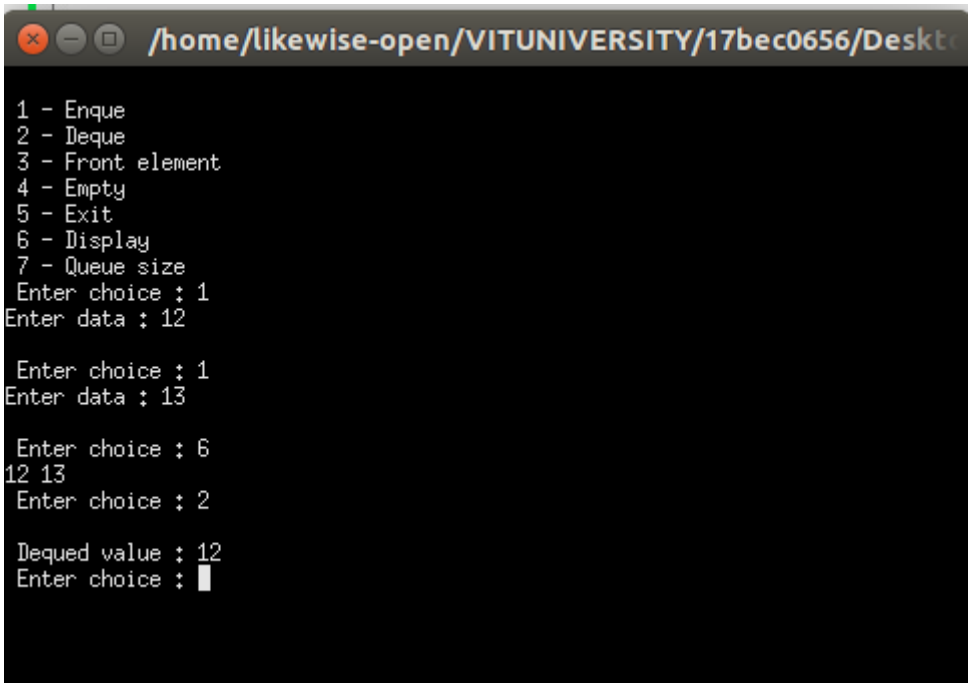
```
#include <stdio.h>
#include <stdlib.h>
 struct node
{
   int info;
   struct node *ptr;
}*front,*rear,*temp,*front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

void main()
{
   int no, ch, e;

   printf("\n 1 - Enque");
   printf("\n 2 - Deque");
   printf("\n 3 - Front element");
   printf("\n 4 - Empty");
   printf("\n 5 - Exit");
   printf("\n 6 - Display");
   printf("\n 7 - Queue size");
   create();
   while (1)
   {
      printf("\n Enter choice : ");
      scanf("%d", &ch);
      switch (ch)
      {
      case 1:
```

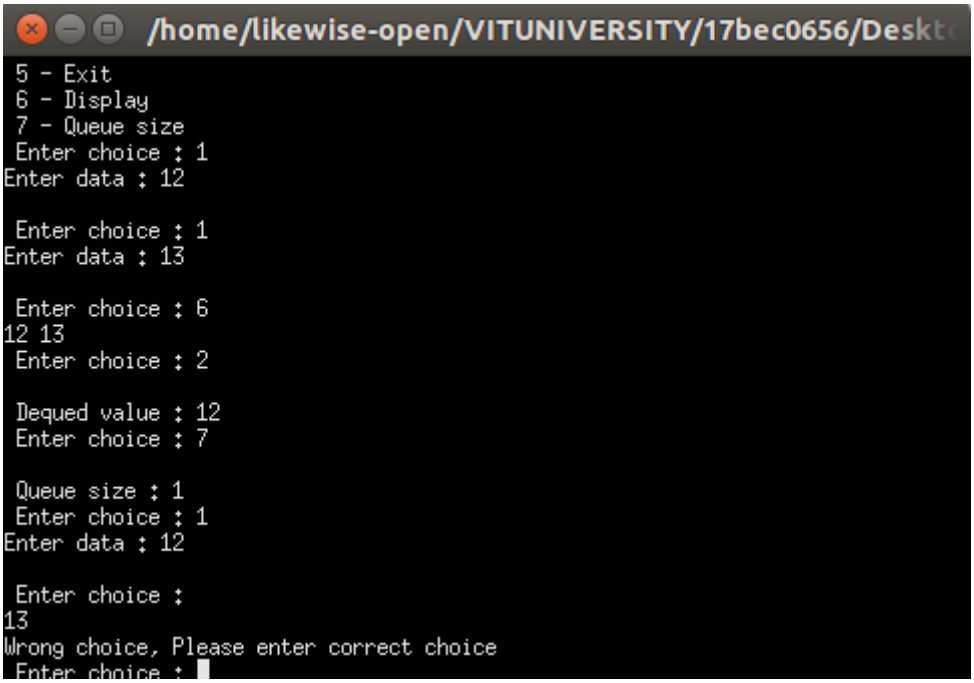
```
            printf("Enter data : ");
            scanf("%d", &no);
            enq(no);
            break;
        case 2:
            deq();
            break;
        case 3:
            e = frontelement();
            if (e != 0)
                printf("Front element : %d", e);
            else
                printf("\n No front element in Queue as queue is empty");
            break;
        case 4:
            empty();
            break;
        case 5:
            exit(0);
        case 6:
            display();
            break;
        case 7:
            queuesize();
            break;
        default:
            printf("Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}

/* Create an empty queue */
void create()
{
    front = rear = NULL;
}

/* Returns queue size */
void queuesize()
{
    printf("\n Queue size : %d", count);
}

/* Enqueing the queue */
void enq(int data)
{
    if (rear == NULL)
    {
```

```
      rear = (struct node *)malloc(1*sizeof(struct node));
      rear->ptr = NULL;
      rear->info = data;
      front = rear;
   }
   else
   {
      temp=(struct node *)malloc(1*sizeof(struct node));
      rear->ptr = temp;
      temp->info = data;
      temp->ptr = NULL;

      rear = temp;
   }
   count++;
}

/* Displaying the queue elements */
void display()
{
   front1 = front;

   if ((front1 == NULL) && (rear == NULL))
   {
      printf("Queue is empty");
      return;
   }
   while (front1 != rear)
   {
      printf("%d ", front1->info);
      front1 = front1->ptr;
   }
   if (front1 == rear)
      printf("%d", front1->info);
}

/* Dequeing the queue */
void deq()
{
   front1 = front;

   if (front1 == NULL)
   {
      printf("\n Error: Trying to display elements from empty queue");
      return;
   }
   else
      if (front1->ptr != NULL)
      {
```

```
        front1 = front1->ptr;
        printf("\n Dequed value : %d", front->info);
        free(front);
        front = front1;
      }
      else
      {
        printf("\n Dequed value : %d", front->info);
        free(front);
        front = NULL;
        rear = NULL;
      }
      count--;
}

/* Returns the front element of queue */
int frontelement()
{
   if ((front != NULL) && (rear != NULL))
      return(front->info);
   else
      return 0;
}

/* Display if queue is empty or not */
void empty()
{
    if ((front == NULL) && (rear == NULL))
       printf("\n Queue empty");
   else
     printf("Queue not empty");
}
```

Program Output:





Question 3

Pseudocode

1. Reading from node

```
read(node *h)
{
```

```
int n,i,j,power,coeff;
node *p;
p=init();
printf("n Enter number of terms :");
scanf("%d",&n);
/* read n terms */
for (i=0;i<n;i++)
{      printf("nenter a term(power  coeff.)");
scanf("%d%d",&power,&coeff);
for(p=h,j=0;j<power;j++)
   p=p->next;
p->coeff=coeff;
}
```

2. Printing

```
print(node *p)
{
  int i;
for(i=0;p!=NULL;i++,p=p->next)
if(p->coeff!=0)
printf("%dX^%d   ",p->coeff,i);
}
```

3. Adding 2 polynomials

```
node * add(node *h1, node *h2)
{
    node *h3,*p;
    h3=init();
    p=h3;
    while(h1!=NULL)
     {
h3->coeff=h1->coeff+h2->coeff;
h1=h1->next;
h2=h2->next;
h3=h3->next;
     }
 return(p);
}
```

4. Multiplying 2 polynomials.

```
 multiply(node *h1, node *h2)
{
node *h3,*p,*q,*r;
int i,j,k,coeff,power;
h3=init();
for(p=h1,i=0;p!=NULL;p=p->next,i++)
for(q=h2,j=0;q!=NULL;q=q->next,j++)
  {
coeff=p->coeff * q->coeff;
```

```
power=i+j;
for(r=h3,k=0;k<power;k++)
r=r->next;
r->coeff=r->coeff+coeff;
   }
  return(h3);
}
```

5. Creation of new node

```
node * init()
{
   int i;
   node *h=NULL,*p;
   for(i=0;i<MAX;i++)
{
p=(node*)malloc(sizeof(node));
p->next=h;
p->coeff=0;
h=p;
}
  return(h);
}
```

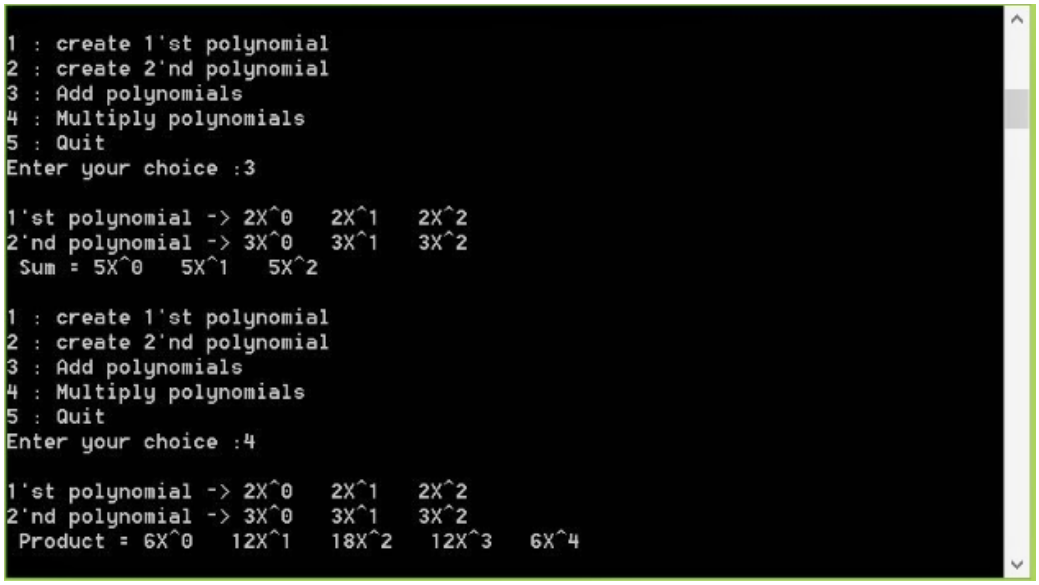
C code:

```
#include<math.h>
#include<stdio.h>
#include<conio.h>
#define MAX 17
typedef struct node
 {
int coeff;
struct node *next;
 }node;
node *  init();
void read(node *h1);
void print(node *h1);
node * add(node *h1,node *h2);
node * multiply(node *h1,node *h2);
/*Polynomial is stored in a linked list, ith node  gives coefficient of x^i .
  a polynomial 3x^2 + 12x^4 will be represented as (0,0,3,0,12,0,0,….)
*/
void main()
{
node *h1=NULL,*h2=NULL,*h3=NULL;
int option;
do
{
printf("n1 : create 1'st polynomial");
```

```
printf(“n2 : create 2’nd polynomial”);
printf(“n3 : Add polynomials”);
printf(“n4 : Multiply polynomials”);
printf(“n5 : Quit”);
printf(“nEnter your choice :”);
scanf(“%d”,&option);
switch(option)
{
case 1:h1=init();read(h1);break;
case 2:h2=init();read(h2);break;
case 3:h3=add(h1,h2);
     printf(“n1’st polynomial -> “);
     print(h1);
     printf(“n2’nd polynomial -> “);
     print(h2);
     printf(“n Sum = “);
     print(h3);
     break;
case 4:h3=multiply(h1,h2);
     printf(“n1’st polynomial -> “);
     print(h1);
     printf(“n2’nd polynomial -> “);
     print(h2);
     printf(“n Product = “);
     print(h3);
     break;
}
}while(option!=5);
}
void  read(node *h)
{
int n,i,j,power,coeff;
node *p;
p=init();
printf(“Enter number of terms :”);
scanf(“%d”,&n);
/* read n terms */
for (i=0;i<n;i++)
{      printf(“enter a term(power  coeff.)”);
scanf(“%d%d”,&power,&coeff);
for(p=h,j=0;j<power;j++)
   p=p->next;
p->coeff=coeff;
}
}
void print(node *p)
{
   int i;
for(i=0;p!=NULL;i++,p=p->next)
```

```
if(p->coeff!=0)
printf("%dX^%d  ",p->coeff,i);
}
node * add(node *h1, node *h2)
{
   node *h3,*p;
   h3=init();
   p=h3;
   while(h1!=NULL)
    {
h3->coeff=h1->coeff+h2->coeff;
h1=h1->next;
h2=h2->next;
h3=h3->next;
    }
 return(p);
}
node * multiply(node *h1, node *h2)
{
node *h3,*p,*q,*r;
int i,j,k,coeff,power;
h3=init();
for(p=h1,i=0;p!=NULL;p=p->next,i++)
for(q=h2,j=0;q!=NULL;q=q->next,j++)
  {
coeff=p->coeff * q->coeff;
power=i+j;
for(r=h3,k=0;k<power;k++)
r=r->next;
r->coeff=r->coeff+coeff;
   }
  return(h3);
}
node * init()
{
   int i;
   node *h=NULL,*p;
   for(i=0;i<MAX;i++)
{
p=(node*)malloc(sizeof(node));
p->next=h;
p->coeff=0;
h=p;
}
  return(h);
}
```

Program Output:


1 : create 1'st polynomial
2 : create 2'nd polynomial
3 : Add polynomials
4 : Multiply polynomials
5 : Quit
Enter your choice :3
1'st polynomial -> 2X^0 2X^1 2X^2
2'nd polynomial -> 3X^0 3X^1 3X^2
Sum = 5X^0 5X^1 5X^2
1 : create 1'st polynomial
2 : create 2'nd polynomial
3 : Add polynomials
4 : Multiply polynomials
5 : Quit
Enter your choice :4
1'st polynomial -> 2X^0 2X^1 2X^2
2'nd polynomial -> 3X^0 3X^1 3X^2
Product = 6X^0 12X^1 18X^2 12X^3 6X^4

Question 4.
Menu driven C program to create binary tree and to perform preorder, inorder and postorder traversal.
Pseudo Code:

1.Inorder

```
inorder(struct node* root){
   if(root == NULL) return;
   inorder(root->left);
   printf("%d ->", root->data);
   inorder(root->right);
}
```

2.

```
void preorder(struct node* root){
   if(root == NULL) return;
   printf("%d ->", root->data);
   preorder(root->left);
   preorder(root->right);
}
```

3.

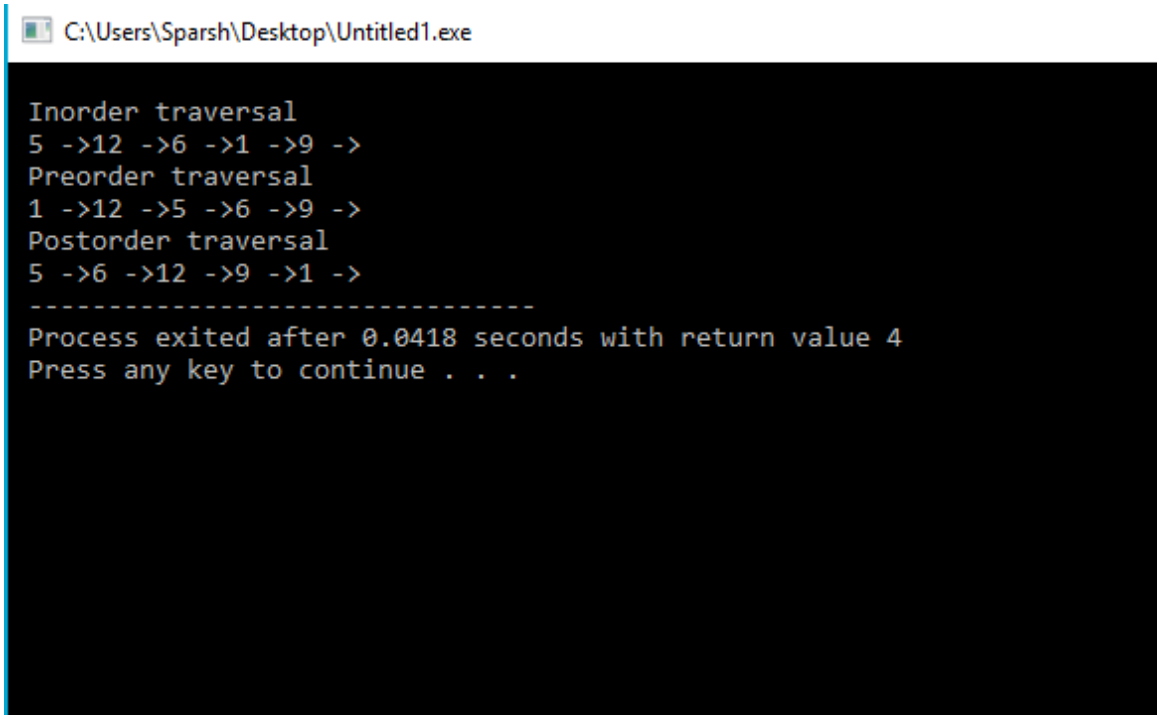
```
postorder(struct node* root) {
   if(root == NULL) return;
   postorder(root->left);
   postorder(root->right);
   printf("%d ->", root->data);
}
```

C code:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
   int data;
   struct node* left;
   struct node* right;
};
void inorder(struct node* root){
   if(root == NULL) return;
   inorder(root->left);
   printf("%d ->", root->data);
   inorder(root->right);
}
void preorder(struct node* root){
   if(root == NULL) return;
   printf("%d ->", root->data);
   preorder(root->left);
   preorder(root->right);
}
```

```
void postorder(struct node* root) {
   if(root == NULL) return;
   postorder(root->left);
   postorder(root->right);
   printf("%d ->", root->data);
}
struct node* createNode(value){
   struct node* newNode = malloc(sizeof(struct node));
   newNode->data = value;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}
struct node* insertLeft(struct node *root, int value) {
   root->left = createNode(value);
   return root->left;
}
struct node* insertRight(struct node *root, int value){
   root->right = createNode(value);
   return root->right;
}
int main(){
   struct node* root = createNode(1);
   insertLeft(root, 12);
   insertRight(root, 9);

   insertLeft(root->left, 5);
   insertRight(root->left, 6);

   printf("Inorder traversal \n");
   inorder(root);
   printf("\nPreorder traversal \n");
   preorder(root);
   printf("\nPostorder traversal \n");
   postorder(root);
}
```

Code Output:


C:\Users\Sparsh\Desktop\Untitled1.exe
Inorder traversal
5 ->12 ->6 ->1 ->9 ->
Preorder traversal
1 ->12 ->5 ->6 ->9 ->
Postorder traversal
5 ->6 ->12 ->9 ->1 ->

Process exited after 0.0418 seconds with return value 4
Press any key to continue . . .