**EX. No.1(a) Array Implementation of LIST ADT**

**Aim**

To perform various operations on List ADT using array implementation.

**Algorithm**

**1.** Start

**2.** Display list of operations as a menu

**3.** Accept user choice

**4.** If choice =1

 Get the number of elements in list

 Create the list of n elements

**5.** If choice = 2 then

Get value of element to be deleted

Move elements one position left wards there on.

Decrement length of the list

Else if choice = 3

Traverse the list and inspect each element

Report position if it exists.

Else if choice = 4

Get position of element to be inserted.

Increment length of the list

Move elements one position right wards there on

Store the new element in corresponding position

**6.** Stop

**Program**

#include<stdio.h>

#include<conio.h>

#define MAX 10

void create();

void insert();

void deletion();

void search();

void display();

void exit();

int a,b[20], n, p, e, f, i, pos,val;

void main()

{

int ch;

char g='y';

printf("\nEnter your name: ");

scanf("%s", name);

printf("\nEnter Register No. ");

scanf("%d", &reg);

do

{

printf("\n main Menu");

printf("\n 1.Create \n 2.Delete \n 3.Search \n 4.Insert \n 5.Display\n  6.Exit \n");

printf("\n Enter your Choice");

scanf("%d", &ch);

switch(ch)

{

case 1:

create();

break;

case 2:

deletion();

break;

case 3:

search();

break;

case 4:

insert();

break;

case 5:

display();

break;

case 6:

exit();

break;

default:

printf("\n Enter the correct choice:"); }

printf("\n Do u want to continue: "); scanf("\n%c", &g);

}

while(g=='y'||g=='Y');

getch();

}

void create()

{

printf("\n Enter the number of nodes"); scanf("%d", &n);

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

{

printf("\n Enter the Element %d :",i+1); scanf("%d", &b[i]);

}

}

void deletion()

{

printf("\n Enter the value u want to delete::"); scanf("%d", &val);

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

{

if(b[i]==val)

{

 for(;i<n;i++)

 {

 b[i]=b[i+1];

 }

 }

}

n--;

printf("\n The Elements after deletion"); for(i=0;i<n;i++)

{

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

}

}

void search()

{

printf("\n Enter the Element to be searched:"); scanf("%d", &e);

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

{

if(b[i]==e)

{

printf("Value is in the %d Position", i+1); }

}

}

void insert()

{

printf("\n Enter the position u need to insert::"); scanf("%d", &pos);

if(pos>=n+1)

{

printf("\n invalid Location::");

}

else

{

for(i=n-1;i>=pos-1;i--)

{

b[i+1]=b[i];

}

printf("\n Enter the element to insert::\n"); scanf("%d",&p);

b[pos-1]=p;

n++;

}

printf("\n The list after insertion::\n");

display();

}

void display()

{

printf("\n The Elements of The list ADT are:"); for(i=0;i<n;i++)

{

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

}

}

**Ex. No. 1 (b) Implementation LIST ADT Using Singly Linked List  Date:**

**Aim**

**To define a singly linked list node and perform operations such as insertions  and**

**deletions dynamically.**

**Algorithm**

**1. Start**

**2. Define single linked list *node* as self referential structure**

**3. Create *Head* node with label = -1 and next = NULL using**

**4. Display menu on list operation**

**5. Accept user choice**

**6. If choice = 1then**

**Locate node after which insertion is to**

**be done Create a new node and get data**

**part**

**Insert new node at appropriate position by manipulating**

**address Else if choice =2**

**Get node's data to be deleted.**

**Locate the node and delink**

**the node Rearrange the links**

**Else**

**Traverse the list from Head node to node which points to null**

**7. Stop**

**Program**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

} \*head;

void main()

{

int ch,i,reg;

char name[20];

void create();

void insert\_beg();

void insert\_end();

void insert\_before();

void insert\_after();

void delete\_beg();

void delete\_end();

void delete\_node();

void display();

void search();

printf("\nEnter your name: ");

scanf("%s", name);

printf("\nEnter Register No. ");

scanf("%d", &reg\_no);

do

{

printf("MENU\n");

printf("Select any option\n1.Create a list\n 2.Insert  at begining\n 3.Insert at end\n 4.Insert before a  given data\n 5.Insert after a given data\n 6.Delete  the first node\n 7.Delete the last node\n 8.Delete  the given node\n9.search a node\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

display();

break;

case 2:

insert\_beg();

display();

break;

case 3:

insert\_end();

display();

break;

case 4:

insert\_before();

display();

break;

case 5:

insert\_after();

display();

break;

case 6:

delete\_beg();

display();

break;

case 7:

delete\_end();

display();

break;

case 8:

delete\_node();

display();

break;

case 9:

search();

break;

 }

printf("\nEnter 0 to continue\n"); scanf("%d",&i);

}while(i==0);

getch();

}

void create()

{

struct node \*ptr,\*temp;

int n,i=1;

printf("Enter the no. of nodes\n");

scanf("%d",&n);

ptr=(struct node\*)malloc(sizeof(struct node)); printf("enter the first node\n");

scanf("%d",&ptr->data);

head=ptr;

ptr->next=NULL;

while(i<n)

{

temp=(struct node\*)malloc(sizeof(struct node)); printf("enter the next node\n");

scanf("%d",&temp->data);

ptr->next=temp;

temp->next=NULL;

ptr=temp;

i++;

}

}

void display()

{

struct node \*ptr,\*temp;

temp=head;

printf("\nThe elements of the list are:\n"); while(temp->next != NULL)

{

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

temp=temp->next;

}

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

}

void insert\_beg()

{

struct node \*ptr,\*temp;

temp=(struct node\*)malloc(sizeof(struct node)); if(head==NULL)

{

head=temp;

printf("\nEnter the data to be inserted\n"); scanf("%d",&temp->data);

temp->next=NULL;

}

else

{

ptr=head;

head=temp;

printf("\nEnter the data to be inserted\n"); scanf("%d",&temp->data);

temp->next=ptr;

}

}

void insert\_end()

{

struct node \*ptr,\*temp;

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

while(temp->next != NULL)

{

temp=temp->next;

}

printf("\nEnter the data to be inserted\n"); scanf("%d",&ptr->data);

ptr->next=NULL;

temp->next=ptr;

}

void insert\_before()

{

struct node \*ptr,\*temp,\*preptr;

int n,val;

printf("\nEnter the data to be inserted"); scanf("%d",&n);

printf("\nEnter the value before which the data to be  inserted");

scanf("%d",&val);

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

while(temp->data!=val)

{

preptr=temp;

temp=temp->next;

}

ptr->data=n;

ptr->next=temp;

preptr->next=ptr;

}

void insert\_after()

{

struct node \*ptr,\*temp;

int n,val;

printf("\nEnter the data to be inserted"); scanf("%d",&n);

printf("\nEnter the value after which the data to be  inserted");

scanf("%d",&val);

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

while(temp->data!=val)

{

temp=temp->next;

}

ptr->data=n;

ptr->next=temp->next;

temp->next=ptr;

}

void delete\_beg()

{

struct node \*ptr,\*temp;

ptr=head;

head=ptr->next;

free(ptr);

}

void delete\_end()

{

struct node \*ptr,\*temp;

ptr=head;

while(ptr->next!=NULL)

{

temp=ptr;

ptr=ptr->next;

}

temp->next=NULL;

free(ptr);

}

void delete\_node()

{

struct node \*ptr,\*temp;

intval;

printf("\nEnter the data to be deleted\n"); scanf("%d",&val);

ptr=head;

while(ptr->data!=val)

{

temp=ptr;

ptr=ptr->next;

}

temp->next=ptr->next;

free(ptr);

}

void search()

{

struct node \*temp;

intval, count=0;

printf("\nEnter the value to be searched\n"); scanf("%d",&val);

temp=head;

while(temp->data!=val)

{

count++;

temp=temp->next;

}

printf("The value is in position %d",count+1); }

**Ex.No.2a Array Implementation of Stack ADT Date:**

**Aim**

To implement stack operations using array.

**Algorithm**

**1.** Start

**2.** Define a array *stack* of size *max* =5

**3.** Initialize *top* =-1

**4.** Display a menu listing stack operations

**5.** Accept choice

**6.** If choice = 1then

If top < max -1

Increment top

Store element at current position of top

Else

Print Stack overflow

Else If choice = 2 then

If top < 0 then

Print Stack underflow

Else

Display current top

element Decrement top

Else If choice = 3 then

Display stack elements starting from top

**7.** Stop

**Program**

#include <stdio.h>

#include <conio.h>

#define max 5

staticint stack[max];

int top = -1;

void push(int x)

{

stack[++top] = x;

}

int pop()

{

return (stack[top--]);

}

void view()

{

inti;

if (top < 0)

printf("\n Stack Empty \n");  else

{

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

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

 {

printf("%4d", stack[i]);

 }

printf("\n");

}

}

int main()

{

intch=0, val;

while(ch != 4)

{

printf("\n STACK OPERATION \n");  printf("1.PUSH ");

printf("2.POP ");

printf("3.Display ");

printf("4.QUIT \n");

printf("Enter Choice : ");

scanf("%d", &ch);

switch(ch)

{

case 1:

if(top < max-1)

{

printf("\nEnter Stack element : ");  scanf("%d", &val);

push(val);

}

else

printf("\n Stack Overflow \n");  break;

case 2:

if(top < 0)

printf("\n Stack Underflow \n");  else

{

val = pop();

printf("\n Popped element is %d\n", val); }

break;

case 3:

view();

break;

case 4:

exit(0);

default:

printf("\n Invalid Choice \n"); }

}

}

**Ex. No. 2b Linked List Implementation of Stack Date:**

**Aim**

**To implement stack operations using linked list.**

**Algorithm**

**1. Start**

**2. Define a singly linked list node for stack**

**3. Create Head node**

**4. Display a menu listing stack operations**

**5. Accept choice**

**6. If choice = 1then**

**Create a new node with data**

**Make new node point to first**

**node**

**Make head node point to new**

**node Else If choice = 2 then**

**Make temp node point to first node**

**Make head node point to next of temp node Release memory Else If choice = 3 then**

**Display stack elements starting from head node till null**

**7. Stop**

**Program**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <malloc.h>

struct stack

{

int data;

struct stack \*next;

};

struct stack \*top = NULL;

struct stack \*push(struct stack \*, int);

struct stack \*display(struct stack \*);

struct stack \*pop(struct stack \*);

int peek(struct stack \*);

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

{

int val, option;

do

{

printf("\n \*\*\*\*\*MAIN MENU\*\*\*\*\*");

printf("\n 1. PUSH");

printf("\n 2. POP");

printf("\n 3. PEEK");

printf("\n 4. DISPLAY");

printf("\n 5. EXIT");

printf("\n Enter your option: ");

scanf("%d", &option);

switch(option)

{

case 1:

printf("\n Enter the number to be pushed on stack: "); scanf("%d", &val);

top = push(top, val);

break;

case 2:

top = pop(top);

break;

case 3:

val = peek(top);

if (val != -1)

printf("\n The value at the top of stack is: %d", val); else

printf("\n STACK IS EMPTY");

break;

case 4:

top = display(top);

break;

}

}while(option != 5);

return 0;

}

struct stack \*push(struct stack \*top, int val) {

struct stack \*ptr;

ptr = (struct stack\*)malloc(sizeof(struct stack)); ptr -> data = val;

if(top == NULL)

{

ptr -> next = NULL;

top = ptr;

}

else

{

ptr -> next = top;

top = ptr;

}

return top;

}

struct stack \*display(struct stack \*top) {

struct stack \*ptr;

ptr = top;

if(top == NULL)

printf("\n STACK IS EMPTY");

else

**{**

while(ptr != NULL)

{

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

ptr = ptr -> next;

}

}

return top;

}

struct stack \*pop(struct stack \*top)

{

struct stack \*ptr;

ptr = top;

if(top == NULL)

printf("\n STACK UNDERFLOW");

else

{

top = top -> next;

printf("\n The value being deleted is: %d", ptr -> data); free(ptr);

}

return top;

}

int peek(struct stack \*top)

{

if(top==NULL)

return -1;

else

return top ->data;

}

**Ex. No. 3(a) Array Implementation of Queue ADT Date:**

**Aim**

To implement queue operations using array.

**Algorithm**

1. Start

2. Define a array queue of size max = 5

3. Initialize front = rear = –1

4. Display a menu listing queue operations

5. Accept choice

6. If choice = 1 then If rear < max -1

Increment rear

Store element at current position of rear

Else

Print Queue Full

Else If choice = 2 then If front = –1 then

Print Queue empty

Else

Display current front element Increment front

Else If choice = 3 then,

Display queue elements starting from front to rear.

7. Stop

**Program**

#include <stdio.h>

#include <conio.h>

#define max 5

staticint queue[max];

int front = -1;

int rear = -1;

void insert(int x)

{

queue[++rear] = x;

if (front == -1)

front = 0;

}

int rem()

{

intval;

val = queue[front];

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

else

front ++;

return (val);

}

void view()

{

inti;

if (front == -1)

printf("\n Queue Empty \n");  else

{

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

for(i=front; i<=rear; i++) printf("%4d", queue[i]);

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

}

}

int main()

{

intch= 0,val;

while(ch != 4)

{

printf("\n QUEUE OPERATION \n");

printf("1.INSERT ");

printf("2.DELETE ");

printf("3.DISPLAY ");

printf("4.QUIT\n");

printf("Enter Choice : ");

scanf("%d", &ch);

switch(ch)

{

case 1:

if(rear < max-1)

{

printf("\n Enter element to be inserted : ");  scanf("%d", &val);

insert(val);

}

else

printf("\n Queue Full \n");

break;

case 2:

if(front == -1)

printf("\n Queue Empty \n");

else

{

val = rem();

printf("\n Element deleted : %d \n", val); }

break;

case 3:

view();

break;

case 4:

exit(0);

default:

printf("\n Invalid Choice \n"); }

}

}

**Ex.No.3 (b) Linked List Implementation of Queue Date:**

**Aim**

**To implement queue operations using linked list.**

**Algorithm**

**1. Start**

**2. Define a singly linked list node for stack**

**3. Create Head node**

**4. Display a menu listing stack operations**

**5. Acceptchoice**

**6. If choice = 1then**

**Create a new node with data Make**

**new node point to first node**

**Make head node point to new node**

**Else If choice = 2 then**

**Make temp node point to first node**

**Makeheadnodepointtonextoftempnode**

**Releasememory**

**Else If choice = 3 then**

**Display stack elements starting from head node till null**

**7. Stop**

**Program**

#include <stdio.h>

#include <malloc.h>

voidenqueue();

voiddequeue();

void display();

struct node

{

int data;

struct node \*next;

}\*front, \*rear;

int main()

{

int option;

struct node \*front;

struct node \*rear;

do

{

printf("\n \*\*\*\*\*QUEUE OPERATION MENU\*\*\*\*\*"); printf("\n 1. INSERT");

printf("\n 2. DELETE");

printf("\n 3. DISPLAY");

printf("\n 4. EXIT");

printf("\n Enter your option : "); scanf("%d", &option);

switch(option)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

}

}while(option != 4);

return 0;

}

void enqueue()

{

struct node \*ptr,\*temp;

temp=(struct node\*)malloc(sizeof(struct node)); printf("\nEnter the data to be inserted\n"); scanf("%d",&temp->data);

if(front == NULL)

{

front=temp;

front->next=NULL;

rear=front;

}

else

{

rear->next=temp;

rear=temp;

rear->next=NULL;

}

}

void dequeue( )

{

intval;

struct node \*ptr;

if (front == NULL)

 {

printf("\nThe Queue is empty");

 }

if (rear==front)

 {

ptr=front;

val=ptr->data;

rear=NULL;

front=NULL;

free(ptr);

 }

else

 {

ptr = front;

val=ptr->data;

front = front -> next;

free(ptr);

 }

printf("\n The item deleted is %d", val);  }

void display( )

{

struct node \*ptr;

ptr = front;

printf("\n The Contents of the Queue are:\n"); printf("Front --->\t");

while(ptr->next != NULL)

 {

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

ptr = ptr->next;

 }

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

printf("\t<--- Rear\n");

}

**Ex.No.4a Application of STACK - Infix To Postfix Conversion Date:**

**Aim**

**To convert infix expression to its postfix form using stack operations.**

**Algorithm**

**1. Start**

**2. Define a array *stack* of size *max* =20**

**3. Initialize *top* =-1**

**4. Read the infix expression character-by-character**

**If character is an operand print it**

**If character is an operator**

**Compare the operator’s priority with the stack[top] operator.**

**If the stack[top]has higher/equal priority than the input**

**operator, Pop it from the stack and print it.**

**Else**

**Push the input operator onto the stack**

**If character is a left parenthesis, then push it onto the stack.**

**If character is a right parenthesis, pop all operators from stack and  print it until a left parenthesis is encountered.**

**Do not print the parenthesis.**

**If character = $ then Pop out all operators, Print them and Stop.**

**Program**

#include <stdio.h>

#include <conio.h>

#include <string.h>

#define MAX 20

int top = -1;

char stack[MAX];

char pop();

void push(char item);

int prcd(char symbol) {

switch(symbol)

{

case '+':

case '-':

return 2; break;

case '\*':

case '/':

return 4; break;

case '^':

case '$':

return 6; break;

case '(':

case ')':

case '#':

return 1; break;

}

}

int isoperator(char symbol) {

switch(symbol)

{

case '+':

case '-':

case '\*':

case '/':

case '^':

case '$':

case '(':

case ')':

return 1;

break; default:

return 0;

}

}

void convertip(char infix[],char postfix[]) {

int i, symbol, j = 0;

stack[++top] = '#';

for(i=0;i < strlen(infix);i++)

{

symbol = infix[i];

if(isoperator(symbol) == 0)

{

postfix[j] = symbol; j++;

}

else

{

if(symbol == '(')

push(symbol);

else if(symbol == ')')

{

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

{

postfix[j] = pop(); j++;

}

pop(); //pop out (.

}

else

{

if(prcd(symbol) >prcd(stack[top]))  push(symbol);

else

{

while(prcd(symbol) <= prcd(stack[top])) {

postfix[j] = pop(); j++;

}

push(symbol);

}

}

}

}

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

{

postfix[j] = pop();

j++;

}

postfix[j] = '\0';

}

main()

{

char infix[20],postfix[20];

printf("Enter the valid infix string: ");  scanf("%s",infix);

printf(“\nEnter Your name: ”);

scanf(“%s”,name);

convertip(infix, postfix);

printf("The corresponding postfix string is: ");  puts(postfix);

}

void push(char item)

{

top++;

stack[top] = item;

}

char pop()

{

char a;

a = stack[top];

top--;

return a;

}

**Ex.No.4b Application of STACK - Postfix Expression Evaluation Date:**

**Aim**

**To evaluate the given postfix expression using stack operations.**

**Algorithm**

**1. Start**

**2. Define a array *stack* of size *max* =20**

**3. Initialize *top* =-1**

**4. Read the postfix expression character-by-character**

**If character is an operand push it on to the stack**

**If character is an operator**

**Pop topmost two elements from stack.**

**Apply operator on the elements and push the result onto the**

**stack,**

**5. Eventually only result will be in the stack at end of the expression. 6. Pop the result and print it.**

**7. Stop**

**Program**

#include <stdio.h>

#include <conio.h>

struct stack

{

int top;

float a[50];

}s;

int top;

float a[50];

main()

{

char pf[50];

float d1,d2,d3;

int i;

s.top = -1;

printf("\n\n Enter the postfix expression: ");  gets(pf);

for(i=0; pf[i]!='\0'; i++)

{

switch(pf[i])

{

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9':

s.a[++s.top] = pf[i]-'0';

break;

case '+':

d1 = s.a[s.top--];

d2 = s.a[s.top--];

s.a[++s.top] = d1 + d2;

break;

case '-':

d2 = s.a[s.top--];

d1 = s.a[s.top--];

s.a[++s.top] = d1 - d2;

break;

case '\*':

d2 = s.a[s.top--];

d1 = s.a[s.top--];

s.a[++s.top] = d1\*d2;

break;

case '/':

d2 = s.a[s.top--];

d1 = s.a[s.top--];

s.a[++s.top] = d1 / d2;

break;

}

}

printf("\n Expression value is %5.2f", s.a[s.top]);  getch();

}

**Ex. No. 4c Polynomial Addition Using Linked List Date:**

**Aim**

To add any two given polynomial using linked lists.

**Algorithm**

1. Create a structure for polynomial with exp and coeff terms. 2. Read the coefficient and exponent of given two polynomials p and q. 3. **While** p and q are not null, repeat step 4.

**If** powers of the two terms are equal then

Insert the sum of the terms into the sum Polynomial Advance p and q **Else if** the power of the first polynomial> power of second then  Insert the term from first polynomial into sum polynomial Advance p **Else**

Insert the term from second polynomial into sum polynomial  Advance q

4. Copy the remaining terms from the non empty polynomial into the sum  polynomial

5. Stop

**Program**

#include <stdio.h>

#include<malloc.h>

#include <conio.h>

struct link

{

int coeff; int pow;

struct link \*next;

};

struct link \*poly1=NULL,\*poly2=NULL,\*poly=NULL;

void create(struct link \*node)

{

int i, term;

printf("\n Enter no. of terms in polynomial"); scanf("%d", &term);

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

{

printf("\nEnter coefficient:");

scanf("%d", &node->coeff);

printf("Enter exponent: ");

scanf("%d", &node->pow);

node->next = (struct link\*)malloc(sizeof(struct link)); node =node->next;

node->next =NULL;

}

}

void show(struct link\*node)

{

while(node->next!=NULL)

{

printf("%dx^%d",node->coeff,node->pow);

node=node->next;

if(node->next!=NULL) printf(" + ");

}

}

void polyadd(struct link \*poly1, struct link \*poly2, struct  link \*poly)

{

while(poly1->next && poly2->next)

{

if(poly1->pow> poly2->pow)

{

poly->pow = poly1->pow;

poly->coeff = poly1->coeff;

poly1 = poly1->next;

}

else if(poly1->pow< poly2->pow)

{

poly->pow = poly2->pow;

poly->coeff = poly2->coeff;

poly2 = poly2->next;

}

else

{

poly->pow = poly1->pow;

poly->coeff=poly1->coeff+poly2->coeff;

poly1 =poly1->next;

poly2 =poly2->next;

}

poly->next=(struct link \*)malloc(sizeof(struct link));  poly=poly->next;

poly->next=NULL;

}

while(poly1->next || poly2->next)

{

if(poly1->next)

{

poly->pow = poly1->pow;

poly->coeff = poly1->coeff;

poly1 = poly1->next;

}

if(poly2->next)

{

poly->pow = poly2->pow; poly->coeff = poly2->coeff;  poly2 = poly2->next;

}

poly->next = (struct link \*)malloc(sizeof(struct link)); poly = poly->next; poly->next = NULL;

}

}

main()

{

poly1 = (struct link \*)malloc(sizeof(struct link));  poly2 = (struct link \*)malloc(sizeof(struct link));  poly = (struct link \*)malloc(sizeof(struct link)); printf("Enter 1stPolynomial:");

create(poly1);

printf("\nEnter 2nd Polynomial:");

create(poly2);

printf("\nPoly1:");

show(poly1);

printf("\n Poly2:");

show(poly2);

polyadd(poly1, poly2, poly);

printf("\n Added Polynomial:");

show(poly);

}

**Ex. No. 5 Binary Tree Traversal Date:**

**Aim**

**To implement different types of tree traversal for the given binary tree.**

**Algorithm**

**1. Create a structure with key and 2 pointer variable left and right 2. Read the node to be inserted.**

**If(root==NULL)**

**root=node**

**else if (root->key < node-**

**>key) root-**

**>right=NULL**

**else**

**Root->left=node**

**3. For Inorder Traversal**

**Traverse Left**

**subtree Visit**

**root**

**Traverse Right subtree**

**4. For Preorder Traversal**

**Visit root**

**Traverse Left**

**subtree Traverse**

**Right subtree**

**5. For Postorder Traversal**

**Traverse Left**

**subtree Traverse**

**Right subtree**

**Visit root**

**6. Stop.**

**Program**

#include <stdio.h>

#include<stdlib.h>

typedef struct node

{

int data;

struct node \*left;

struct node\*right;

}node;

int count=1;

node \*insert(node \*tree, int digit) {

if(tree == NULL)

{

tree=(node\*)malloc(sizeof(node));  tree->left =tree->right=NULL;

tree->data =digit; count++;

}

else if(count%2 == 0)

tree->left=insert(tree->left,digit);  else

tree->right=insert(tree->right,digit);  return tree;

}

void preorder(node \*t)

{

if(t != NULL)

{

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

preorder(t->left);

preorder(t->right);

}

}

void postorder(node \*t)

{

if(t != NULL)

{

postorder(t->left);

postorder(t->right);

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

}

}

void inorder(node \*t)

{

if(t != NULL)

{

inorder(t->left);

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

inorder(t->right);

}

}

void main()

{

node \*root =NULL;

int i,n, digit;

printf("\n Enter the number of elements in the tree:  ");

scanf("%d",&n);

printf("Enther the %d Elements one by one", n); for(i=0;i<n;i++)

{

scanf("%d",&digit);

root=insert(root,digit);

}

printf("\n The pre order traversal of the tree is:\n  ");

preorder(root);

printf("\n The inorder traversal of tree is:\n ");  inorder(root);

printf("\n The post order traversal of tree is:\n "); postorder(root);

}