ARRAY QUEUE

#include<stdio.h>

int Front=-1, Rear=-1;

#define MAX 20

int Queue[MAX];

void insert()

{

int val;

printf("Enter the value to be inserted ");

scanf("%d",&val);

if(Rear==MAX-1)

{

printf("Overflow");

}

else if(Rear==-1)

{

Front=Rear=0;

Queue[Rear]=val;

}

else

{

Rear++;

Queue[Rear]=val;

}

}

void deletion()

{

if((Front>Rear)||(Rear==-1))

{

printf("Underflow");

return;

}

else

{

Front=Front+1;

}

}

void display()

{

int i;

if (Front == - 1)

printf("Queue is empty \n");

else

{

printf("Queue is : \n");

for (i = Front; i <= Rear; i++)

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

printf("\n");

}

}

int main()

{

int op, i, n;

printf("Array implementation of Queue \n");

printf("Enter 1 to insert\n; 2 to Delete;\n 3 to display\n 0 to Exit");

scanf("%d",&op);

while(op!=0)

{

switch(op)

{

case 1:

printf("Enter the number of elements to insert");

scanf("%d",&n);

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

{

insert();

}

break;

case 2:

deletion();

break;

case 3:

display();

break;

}

printf("Enter 1 to insert\n; 2 to Delete;\n 3 to display\n 0 to Exit");

scanf("%d",&op);

}

}

ARRAY STACK

#include<stdio.h>

int Top=-1;

#define MAX 20

int Stack[MAX];

void push()

{

int val;

printf("Enter the value to be pushed ");

scanf("%d",&val);

if(Top==MAX-1)

{

printf("Overflow");

}

else if(Top==-1)

{

Top=0;

Stack[Top]=val;

}

else

{

Top=Top+1;

Stack[Top]=val;

}

}

void pop()

{

if(Top==-1)

{

printf("Underflow");

return;

}

else

{

Top=Top-1;

}

}

void peek()

{

if(Top==-1)

{

printf("Underflow");

return;

}

else

{

printf("Stack Top is %d", Stack[Top]);

}

}

void display()

{

int i;

if (Top == -1)

printf("Stack is empty \n");

else

{

printf("Stack is : \n");

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

{printf("%d ", Stack[i]);

printf("\n");}

}

}

int main()

{

int op, i, n;

printf("Array implementation of Stack \n");

printf("Enter 1 to push\n; 2 to pop;\n 4 to display\n 3 To Peek\n 0 to Exit");

scanf("%d",&op);

while(op!=0)

{

switch(op)

{

case 1:

printf("Enter the number of elements to push");

scanf("%d",&n);

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

{

push();

}

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display();

break;

}

printf("Enter 1 to push\n; 2 to pop;\n 4 to display\n 3 To Peek\n 0 to Exit");

scanf("%d",&op);

}

return 0;

}

LINKED LIST QUEUE

#include<stdio.h>

struct node

{

struct node \*next;

int data;

};

struct node \*Front=NULL, \*Rear=NULL;

void insert()

{

struct node \*New\_node;

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

printf("Enter the data ");

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

if(Front==NULL)

{

Front=Rear=New\_node;

New\_node->next=NULL;

}

else

{Rear->next= New\_node;

Rear=New\_node;

New\_node->next=NULL;

}

}

void delete()

{

struct node \*ptr;

if(Front==NULL)

{

printf("No elements in the queue ");

}

else

{ ptr=Front;

Front=Front->next;

free(ptr);

}

}

void display()

{

struct node \*ptr;

printf("The queue is ");

ptr=Front;

while(ptr!=Rear)

{

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

ptr= ptr->next;

}

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

}

int main()

{

int op, i, n;

printf("Linked List implementation of Queue \n");

printf("Enter 1 to insert\n; 2 to Delete;\n 3 to display\n 0 to Exit");

scanf("%d",&op);

while(op!=0)

{

switch(op)

{

case 1:

printf("Enter the number of elements to insert");

scanf("%d",&n);

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

{

insert();

}

break;

case 2:

delete();

break;

case 3:

display();

break;

}

printf("Enter 1 to insert\n; 2 to Delete;\n 3 to display\n 0 to Exit");

scanf("%d",&op);

}

}

LINKED LIST STACK

#include<stdio.h>

struct node

{

struct node \*next;

int data;

};

struct node \*Top=NULL;

void push()

{

struct node \*New\_node, \*ptr;

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

printf("Enter the data ");

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

if(Top==NULL)

{

Top=New\_node;

New\_node->next=NULL;

}

else

{New\_node->next=Top;

Top=New\_node;

}

}

void pop()

{

struct node \*ptr;

if(Top==NULL)

{

printf("No elements in the Stack ");

}

else

{ ptr=Top;

Top=Top->next;

free(ptr);

}

}

void peek()

{

struct node \*ptr;

if(Top==NULL)

{

printf("No elements in the Stack ");

}

else

{ printf("The Top element is %d ", Top->data);

}

}

void display()

{

struct node \*ptr;

printf("The Stack is ");

ptr=Top;

while(ptr->next!=NULL)

{

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

ptr= ptr->next;

}

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

}

int main()

{

int op, i, n;

printf("Linked List implementation of Stack \n");

printf("Enter 1 to push\n; 2 to pop;\n 3 to peek\n4 to display\n 0 to Exit");

scanf("%d",&op);

while(op!=0)

{

switch(op)

{

case 1:

printf("Enter the number of elements to push");

scanf("%d",&n);

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

{

push();

}

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display();

break;

}

printf("Enter 1 to push\n; 2 to pop;\n 3 to peek\n4 to display\n 0 to Exit");

scanf("%d",&op);

}

}

BST

#include<stdio.h>

struct btnode

{

struct btnode \*left;

struct btnode \*right;

int data;

};

struct btnode \*root=NULL;

struct btnode \*temp=NULL, \*temp1, \*node, \*ptr;

int c;

void insert()

{ create();

if(root==NULL)

root=node;

else

{temp=node;

search(root);

}

}

void create()

{

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

printf("Enter the value of the node ");

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

node->left=NULL;

node->right=NULL;

}

void search(struct btnode \*t)

{

if((temp->data<=t->data)&&(t->left!=NULL))

{

search(root->left);

}

else if((temp->data<=t->data)&&(t->left==NULL))

{

t->left=temp;

}

else if((temp->data>t->data)&&(t->right!=NULL))

{

search(t->right);

}

else if((temp->data>t->data)&&(t->right==NULL))

{

t->right=temp;

}

}

void inorder(struct btnode \*t)

{

if(root==NULL)

{

printf("No nodes present in the tree ");

return;

}

else

{

if(t->left!=NULL)

{inorder(t->left);}

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

if(t->right!=NULL)

inorder(t->right);

}

}

void preorder(struct btnode \*t)

{

if(root==NULL)

{

printf("No nodes present in the tree ");

return;

}

else

{

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

if(t->left!=NULL)

{preorder(t->left);}

if(t->right!=NULL)

preorder(t->right);

}

}

void postorder(struct btnode \*t)

{

if(root==NULL)

{

printf("No nodes present in the tree ");

return;

}

else

{

if(t->left!=NULL)

{postorder(t->left);}

if(t->right!=NULL)

postorder(t->right);

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

}

}

void search1(struct btnode \*t)

{

if((t->left==NULL)&&(t->right==NULL))

{

temp1=t;

}

else if((t->left==NULL)&&(t->right!=NULL))

{ ptr=t;

c=1;

temp1=t->right;

search1(temp1);

}

else if((t->left!=NULL)&&(t->right==NULL))

{

ptr=t;

c=0;

temp1=t->left;

search1(temp1);

}

else

{

ptr=t;

c=0;

temp1=t->left;

search1(temp1);

}

}

void deletion(struct node \*t)

{

search1(t);

free(temp1);

if(c==0)

ptr->left=NULL;

else

ptr->right=NULL;

}

int main()

{

int i;

printf(" Binary Search Tree ");

printf("Enter 6 elements ");

for(i=0; i<6; i++)

{

insert();

}

printf("INORDER TRAVERSAL ");

inorder(root);

printf("Deletion ");

deletion(root);

printf("INORDER TRAVERSAL ");

inorder(root);

printf("PREORDER TRAVERSAL ");

preorder(root);

printf("POSTORDER TRAVERSAL ");

postorder(root);

return 0;

}