**DATE-21-09-2022**

**12.STACK APPLICATIONS**

#include<stdio.h>

void TOH(int n,char y,char x,char z)

{if(n>0)

{

TOH(n-1,x,z,y);

printf("\n %c to %c",x,y);

TOH(n-1,z,y,x);

}

}

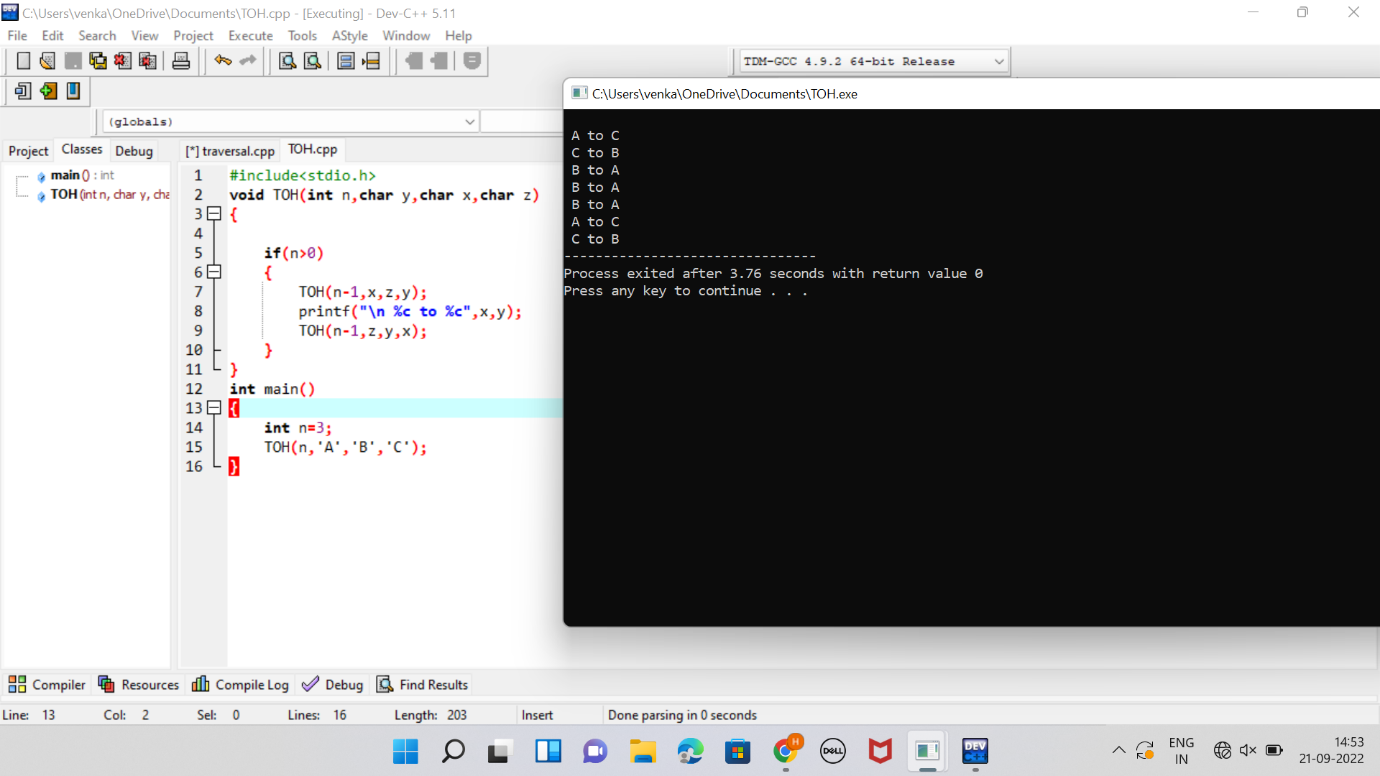
int main()

{

int n=3;

TOH(n,'A','B','C');

}



**13.QUEUE OPERATIONS**

#include<stdio.h>

#include<process.h>

#include<stdlib.h>

#define MAX 5

int front=-1,rear=-1,queue[MAX];

void enqueue();

void dequeue();

void display();

int main()

{

int ch;

while(1)

{

printf("\n\n1.enqueue\n2.dequeue\n3.Display\n4.Exit");

printf("\n\nEnter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{

case 1: enqueue();

break;

case 2: dequeue();

break;

case 3: display();

break;

case 4: exit(0);

default: printf("\nWrong Choice!!");

}

}

}

void enqueue()

{

int val;

if(rear==MAX-1)

{

printf("\nqueue is full!!");

}

else

{

printf("\nEnter element to enqueue:");

scanf("%d",&val);

front=front+1;

queue[front]=val;

}

}

void dequeue()

{

if(front==-1)

{

printf("\nqueue is empty!!");

}

else

{

printf("\nDeleted element is %d",queue[rear]);

front=front-1;

}

}

void display()

{

int i;

if(front==-1)

{

printf("\nqueue is empty!!");

}

else

{

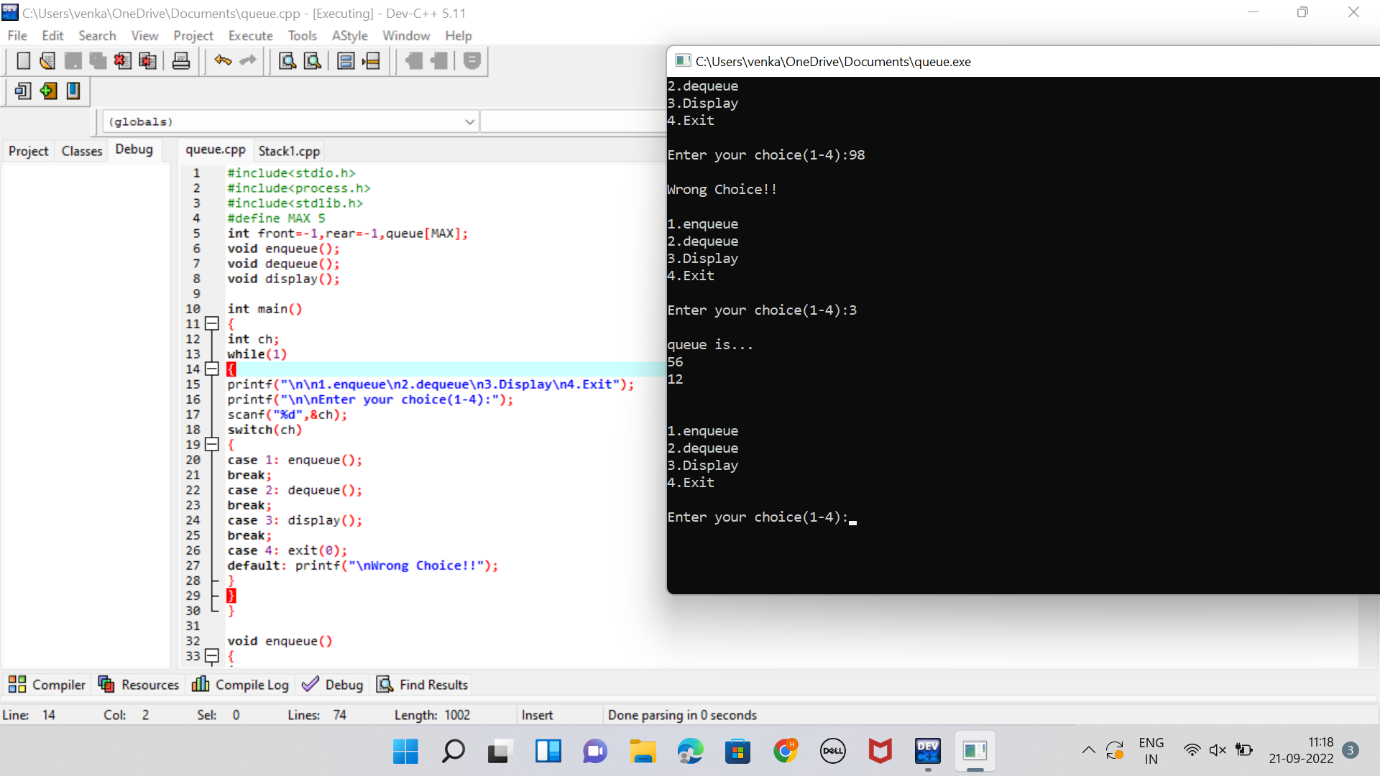
printf("\nqueue is...\n");

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

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

}

}



**14.TRAVERSAL OF BINARY SEARCH TREE**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

struct node\* newNode(int data)

{

struct node\* node = (struct node\*)

malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return(node);

}

void printPostorder(struct node\* node)

{

if (node == NULL)

return;

printPostorder(node->left);

printPostorder(node->right);

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

}

void printInorder(struct node\* node)

{

if (node == NULL)

return;

printInorder(node->left);

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

printInorder(node->right);

}

void printPreorder(struct node\* node)

{

if (node == NULL)

return;

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

printPreorder(node->left);

printPreorder(node->right);

}

int main()

{

struct node \*root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

printf("\nPreorder traversal of binary tree is \n");

printPreorder(root);

printf("\nInorder traversal of binary tree is \n");

printInorder(root);

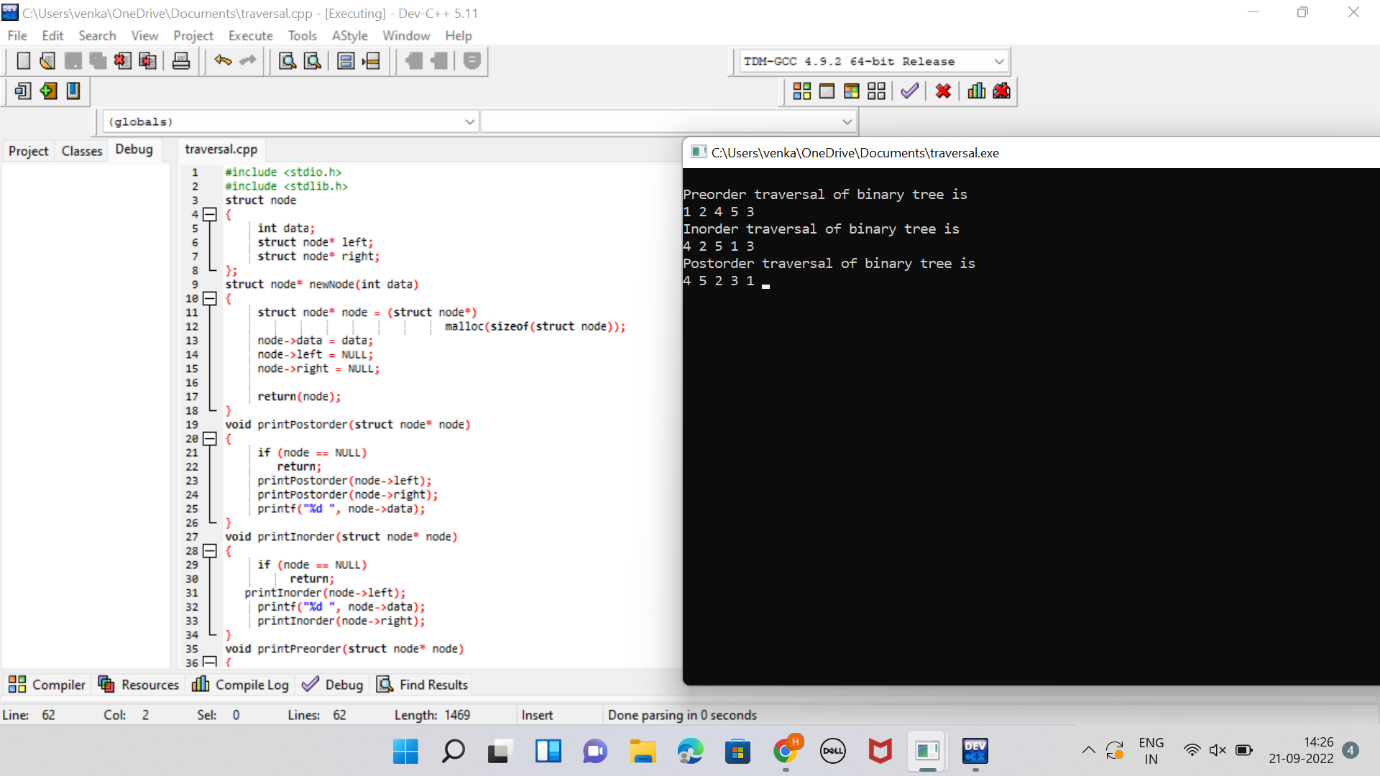
printf("\nPostorder traversal of binary tree is \n");

printPostorder(root);

getchar();

return 0;

}



**15.LINEAR PROBING**

#include<stdio.h>

#include <stdio.h>

#include<stdlib.h>

#define TABLE\_SIZE 10

int h[TABLE\_SIZE]={NULL};

void insert()

{

int key,index,i,flag=0,hkey;

printf("\nenter a value to insert into hash table\n");

scanf("%d",&key);

hkey=key%TABLE\_SIZE;

for(i=0;i<TABLE\_SIZE;i++)

{

index=(hkey+i)%TABLE\_SIZE;

if(h[index] == NULL)

{

h[index]=key;

break;

}

}

if(i == TABLE\_SIZE)

printf("\nelement cannot be inserted\n");

}

void search()

{

int key,index,i,flag=0,hkey;

printf("\nenter search element\n");

scanf("%d",&key);

hkey=key%TABLE\_SIZE;

for(i=0;i<TABLE\_SIZE; i++)

{

index=(hkey+i)%TABLE\_SIZE;

if(h[index]==key)

{

printf("value is found at index %d",index);

break;

}

}

if(i == TABLE\_SIZE)

printf("\n value is not found\n");

}

void display()

{

int i;

printf("\nelements in the hash table are \n");

for(i=0;i< TABLE\_SIZE; i++)

printf("\nat index %d \t value = %d",i,h[i]);

}

main()

{

int opt,i;

while(1)

{

printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");

scanf("%d",&opt);

switch(opt)

{

case 1:

insert();

break;

case 2:

display();

break;

case 3:

search();

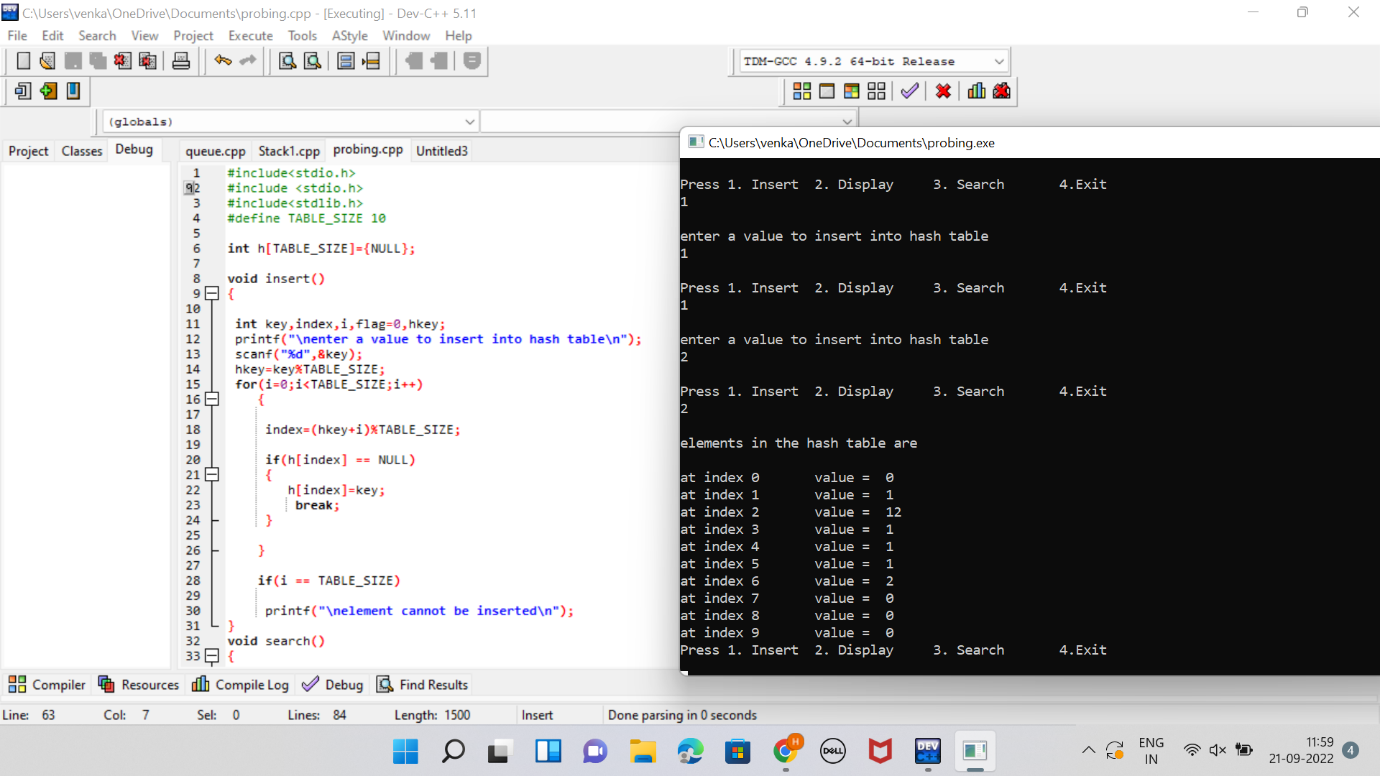
break;

case 4:exit(0);

}

}

}



**16.INSERTION SORT**

#include<stdio.h>

#include<math.h>

void insertionsort(int arr[],int n)

{

int i,key,j;

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

{

key=arr[i];

j=i-1;

while(j>=0&&arr[j]>key)

{

arr[j+1]=arr[j];

j=j-1;

}

arr[j+1]=key;

}}

void printarray(int arr[],int n)

{

int i;

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

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

printf("\n");

}

int main()

{

int arr[]={13,54,1,98,543,89999};

int n=sizeof(arr)/sizeof(arr[0]);

insertionsort(arr,n);

printarray(arr,n);

return 0;

}

