1.1

#include <bits/stdc++.h>

using namespace std;

int arr[] = {1,2,3,4,5};

int main()

{

cout<<"Please Insert two Number : "<<endl;

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

for(int i=len; i <len+2; ++i){

cin>>arr[i];

}

cout<<"Afer Insertion array : ";

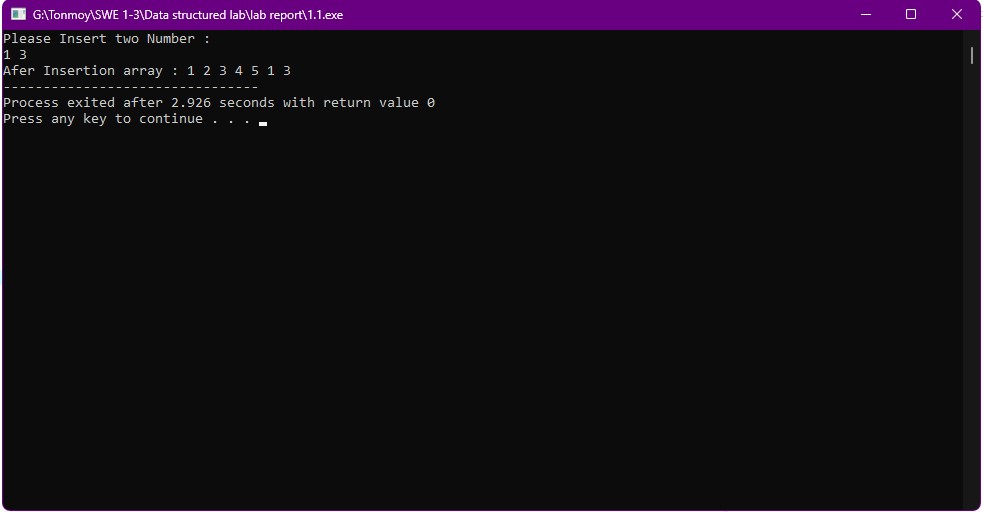
for(int i=0; i < len+2; ++i){

cout<<arr[i]<<" ";

}

return 0;

}



1.2

#include <bits/stdc++.h>

using namespace std;

int arr[] = {1,2,3,4,5};

int main()

{

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

// First Insertion

int posi;

cout<<"insert Position: ";

cin>>posi;

for (int i = len+1; i >= posi ; --i)

{

arr[i] = arr[i-1];

}

len++;

cout<<"Insert Number : ";

cin>>arr[posi-1];

cout<<"Afer Insertion array : ";

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

cout<<arr[i]<<" ";

}

cout<<endl;

// Second Insertion

cout<<"insert Position: ";

cin>>posi;

for (int i = len+1; i >= posi ; --i)

{

arr[i] = arr[i-1];

}

len++;

cout<<"Insert Number : ";

cin>>arr[posi-1];

cout<<"Afer Insertion array : ";

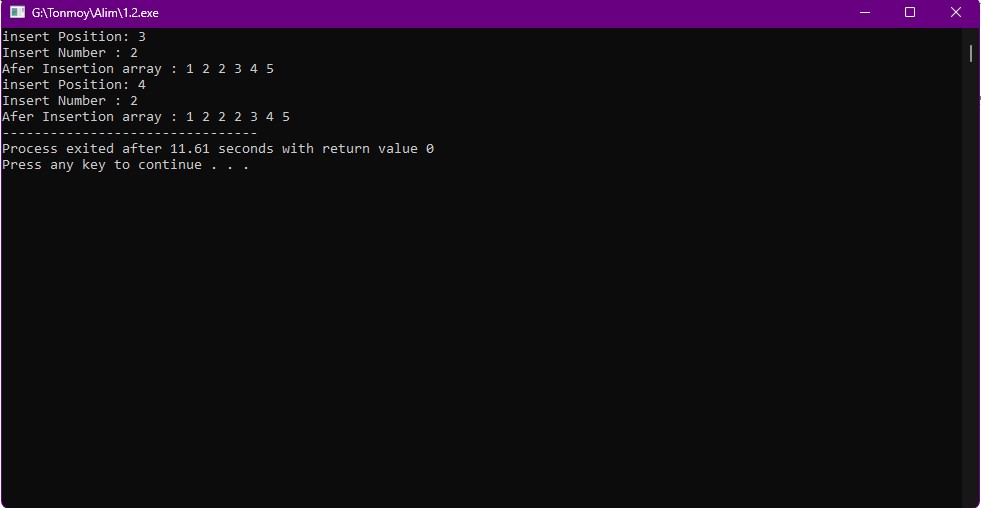
for(int i=0; i < len; ++i){

cout<<arr[i]<<" ";

}

return 0;

}



2

//Suppose you have an array with some elements. Now Find out two elements and delete them if they exist. If not exist any one or both of them, then print "This element can not be deleted as it does not exist".

#include <bits/stdc++.h>

using namespace std;

int arr[] = {1,2,3,4,5};

int main()

{

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

int find;

cout<<"Find first element : ";

cin>>find;

int t = 0;

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

if(arr[i] == find){

++t;

for (int j = i; j < len-1; ++j)

{

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

}

}

}

if(t > 0) cout<<"This element can be deleted as it exist"<<endl;

else cout<<"This element can not be deleted as it does not exist"<<endl;

t = 0;

--len;

cout<<"Find second element : ";

cin>>find;

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

if(arr[i] == find){

t++;

for (int j = i; j < len-1; ++j)

{

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

}

}

}

if(t > 0) cout<<"This element can be deleted as it exist"<<endl;

else cout<<"This element can not be deleted as it does not exist"<<endl;

--len;

cout<<"Afer Insertion array : ";

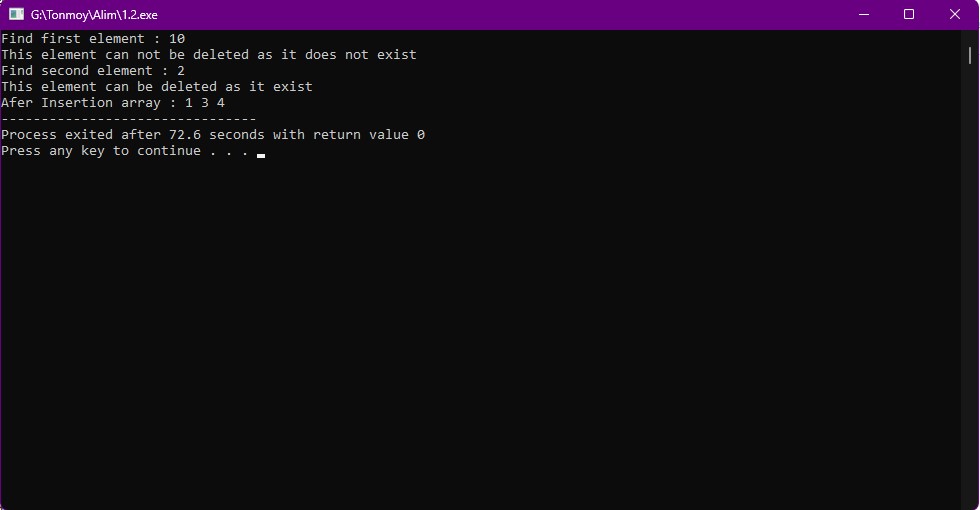
for(int i=0; i < len; ++i){

cout<<arr[i]<<" ";

}

return 0;

}



//Apply binary search to find out an element and delete the element.

#include <bits/stdc++.h>

using namespace std;

int binarySearch(int ar[],int len, int find){

int left = 0;

int right = len-1;

while(left <= right){

int mid = (left+right) / 2;

if(ar[mid] == find){

return mid;

}

if(ar[mid] > find){

right = mid - 1;

}

else left = mid+1;

}

return -1;

}

int main()

{

int arr[] = {1,2,3,4,5};

int find;

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

sort(arr,arr+len);

cout<<"Please find element : ";

cin>>find;

int rtrn = binarySearch(arr,len,find);

if(rtrn > -1){

for(int i=rtrn; i < len-1 ; ++i){

arr[i] = arr[i+1];

}

--len;

}

else{

cout<<"Does not find element .";

}

cout<<"Now Array Element with Delected Element : ";

for (int i = 0; i < len; i++)

{

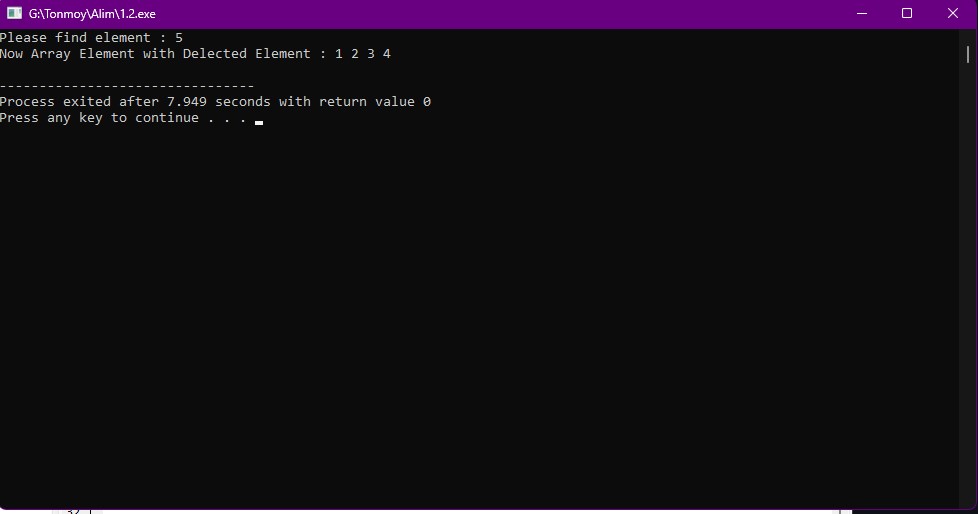
cout<<arr[i] << " ";

}

cout<<endl;

return 0;

}



4

\

#include <stdio.h>

int cnt = 0;

int binarySearch(int arr[], int l, int r, int x)

{

cnt++;

if (r >= l)

{

int mid = (l + r) / 2;

if (arr[mid] == x)return cnt;

if (arr[mid] > x)

return binarySearch(arr, l, mid - 1, x);

return binarySearch(arr, mid + 1, r, x);

}

}

int LinearSearch(int arr[], int n, int x)

{

int count = 0;

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

{

count++;

if (arr[i] == x) return count;

}

}

int main()

{

int x, arr[] = { 1, 2, 3, 5, 6, 7, 9, 10, 12, 14 };

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

printf("Here Is Our Element For Binary and Linear Search : \n");

for (int i = 0; i < n; i++) printf("%d ",arr[i]);

printf("\n");

printf("Enter An Element From Given Array : ");

scanf("%d",&x);

int binary = binarySearch(arr, 0, n - 1, x);

int linear = LinearSearch(arr, n, x);

 printf("Element Found through Binary Search In %d Steps",binary);

printf("Element Found through Linear Search In %d Steps",linear);

printf("\n");

if (linear >= binary)

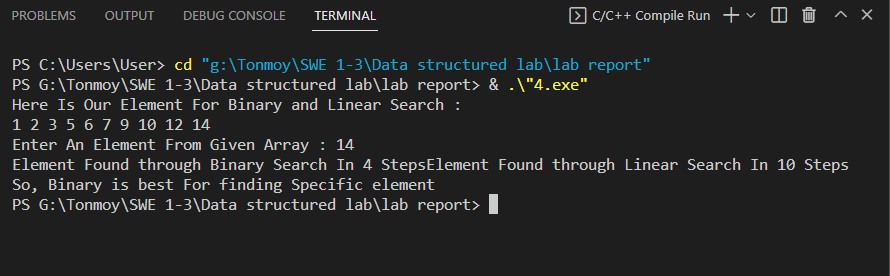
 printf("So, Binary is best For finding Specific element\n");

else

printf("So, Linear is best For finding Specific element\n");

 return 0;

}



5

#include <stdio.h>

int binarySearch(char arr[], int l, int r, char x)

{

if (r >= l)

{

int mid = (l + r) / 2;

if (arr[mid] == x)

return mid;

if (arr[mid] < x)

return binarySearch(arr, l, mid - 1, x);

return binarySearch(arr, mid + 1, r, x);

}

return -1;

}

void SortDecending(char arr[], int n)

{

int temp;

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

{

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

{

if (arr[j] < arr[j + 1])

{

char temp = arr[j + 1];

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

arr[j] = temp;

}

}

}

}

int main()

{

char x, arr[] = { 'B', 'A', 'H', 'N', 'S', 'I', 'L', 'D' };

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

printf("Before Sorting :\n");

for (int i = 0; i < n; i++) printf("%c ",arr[i]);

printf("\n");

SortDecending(arr, n);

printf("After Sorting in Desending Order :\n");

for (int i = 0; i < n; i++) printf("%c \n",arr[i]);

printf("Enter a Character To Search : ");

scanf("%c",&x);

x = toupper(x);

int result = binarySearch(arr, 0, n - 1, x);

if (result == -1) printf("Character is not present in array");

else printf("Found in Position :%d\n",result + 1);

return 0;

}



5.2

//Write down a C program to Sort all the data in Descending Order

#include <bits/stdc++.h>

using namespace std;

int main()

{

int len;

cout<<"Input Your Data length : "<<endl;

cin>>len;

int arr[len];

cout<<"Input Your Data :"<<endl;

for (int i = 0; i < len; i++)

{

cin>>arr[i];

}

for (int i = 0; i < len; i++)

{

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

{

if(arr[j] < arr[j+1]){

int temp = arr[j+1];

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

arr[j] = temp;

}

}

}

cout<<"Sort all data decending order : ";

for (int i = 0; i < len; i++)

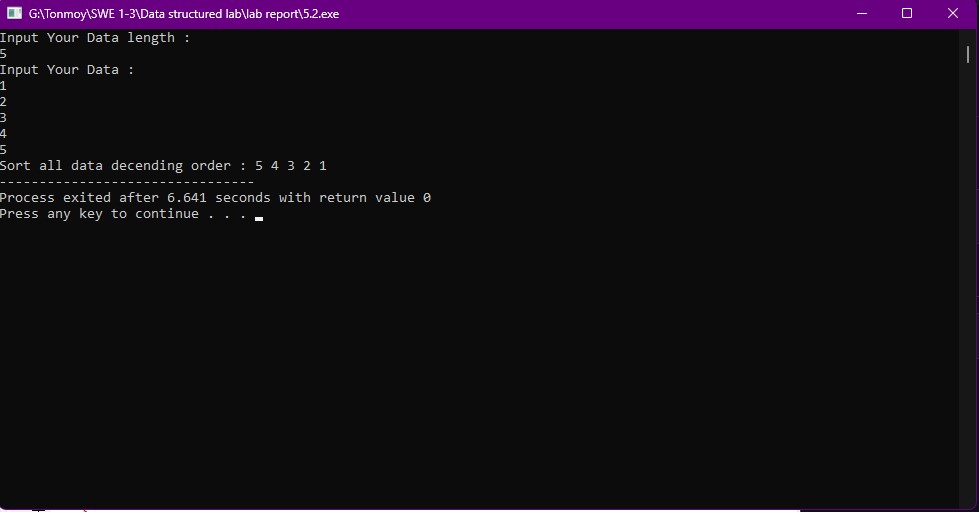
{

cout<<arr[i]<<" ";

}

return 0;

}



6

#include <stdio.h>

#include <string.h>

#define STACK\_MAX 5

int top;

char str[STACK\_MAX+1];

void push(int item)

{

    if (top < STACK\_MAX)

    {

        str[top] = item;

        top = top + 1;

    }

    else

    {

        printf("Don't overflow\n");

    }

}

void pop()

{

    if (top == 0)

    {

        printf("Don't underflow");

        return ;

    }

    else

    {

        top = top - 1;

    }

    return;

}

int main()

{

    top = 0;

    char ch;

    ch = 'o';

    push(ch);

    ch = 'g';

    push(ch);

    ch = 'n';

    push(ch);

    pop();

    ch = 'n';

    push(ch);

    ch = 'a';

    push(ch);

    pop();

    ch = 'a';

    push(ch);

    ch = 'm';

    push(ch);

    pop();

    ch = 'm';

    push(ch);

    for (int i = STACK\_MAX - 1; i >= 0; i--)

    {

        printf("%c", str[i]);

    }

    printf("\n");

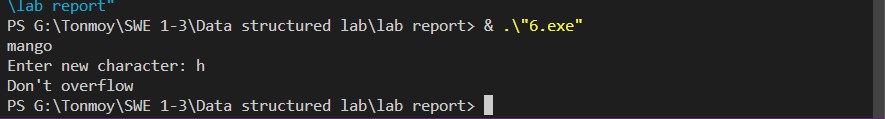
    printf("Enter new character: ");

    scanf("%c",&ch);

    push(ch);

    return 0;

}



7

//Create a linked list. Already it is given in the course materials. Now insert one new element in last position and print the all data of the linked list.

#include <bits/stdc++.h>

using namespace std;

struct Node {

int data;

struct Node\* next;

};

int main()

{

struct Node\* head = NULL;

struct Node\* second = NULL;

struct Node\* third = NULL;

struct Node\* fourth = NULL;

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

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

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

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

head->data = 10;

head->next = second;

second->data = 20;

second->next = third;

third->data = 30;

third->next = fourth;

int i\_list;

cout<<"Insert a new element : "<<endl;

cin>>i\_list;

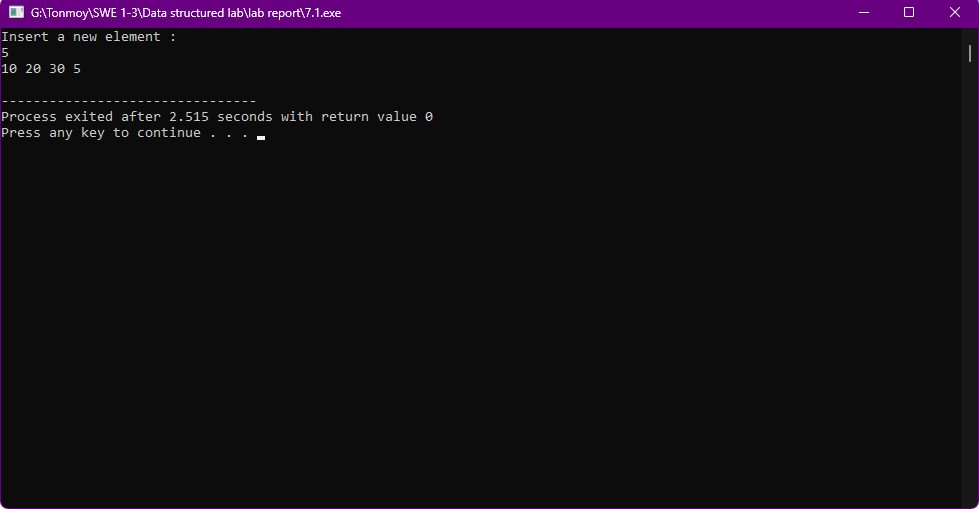
fourth->data = i\_list;

fourth->next = NULL;

cout<<head->data<<" "<<second->data<<" "<<third->data<<" "<<fourth->data<<endl;

return 0;

}



8.1

/\*

\* C Program to Implement Queue Data Structure using Linked List

\*/

#include <bits/stdc++.h>

struct node {

int data;

struct node \*next;

} \*front, \*back;

void initialize() {

front = back = NULL;

}

int getQueueSize() {

struct node \*temp = front;

int count = 0;

if(front == NULL && back == NULL)

return 0;

while(temp != back){

count++;

temp = temp->next;

}

if(temp == back)

count++;

return count;

}

int getFrontElement() {

return front->data;

}

int getBackElement() {

return back->data;

}

void isEmpty() {

if (front == NULL && back == NULL)

printf("Empty Queue\n");

else

printf("Queue is not Empty\n");

}

void enqueue(int num) {

struct node \*temp;

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

temp->data = num;

temp->next = NULL;

if (back == NULL) {

front = back = temp;

} else {

back->next = temp;

back = temp;

}

}

void dequeue() {

struct node \*temp;

if (front == NULL) {

printf("\nQueue is Empty \n");

return;

} else {

temp = front;

front = front->next;

if(front == NULL){

back = NULL;

}

printf("Removed Element : %d\n", temp->data);

free(temp);

}

}

void printQueue() {

struct node \*temp = front;

if ((front == NULL) && (back == NULL)) {

printf("Queue is Empty\n");

return;

}

while (temp != NULL) {

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

temp = temp->next;

if(temp != NULL)

printf("-->");

}

}

int main() {

initialize();

enqueue(1);

enqueue(3);

enqueue(7);

enqueue(5);

enqueue(10);

printQueue();

printf("\nSize of Queue : %d\n", getQueueSize());

printf("Front Element : %d\n", getFrontElement());

printf("Rear Element : %d\n", getBackElement());

dequeue();

dequeue();

dequeue();

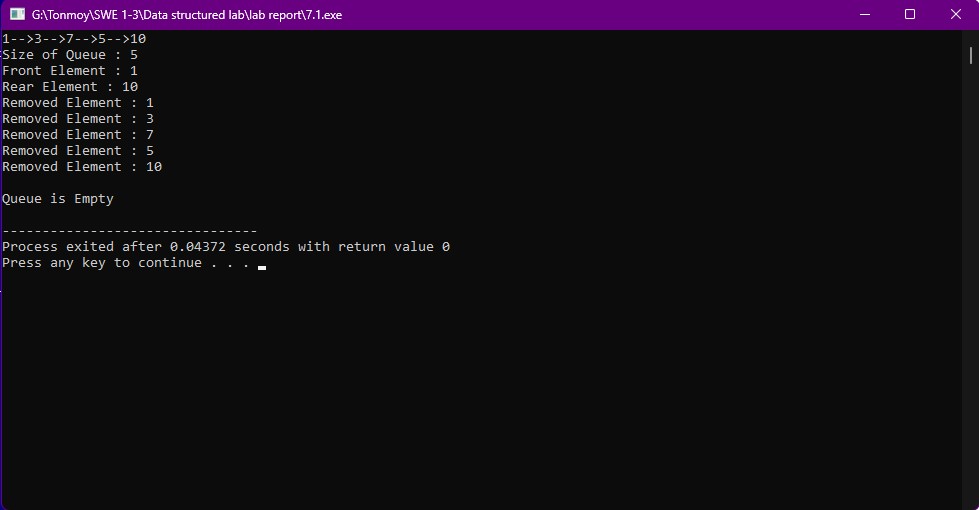
dequeue();

dequeue();

dequeue();

return 0;

}



9.1

#include<bits/stdc++.h>

using namespace std;

struct node

{

char key;

struct node \*left, \*right;

};

struct node \*newNode(int item)

{

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

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

void inorder(struct node \*root)

{

if (root != NULL)

{

inorder(root->left);

cout<<root->key<<endl;

inorder(root->right);

}

}

void print\_preorder(struct node \* root)

{

if (root)

{

cout<<root->key<<endl;

print\_preorder(root->left);

print\_preorder(root->right);

}

}

void print\_postorder(struct node \* root)

{

if (root)

{

print\_postorder(root->left);

print\_postorder(root->right);

cout<<root->key<<endl;

}

}

struct node\* insert(struct node\* node, int key)

{

if (node == NULL) return newNode(key);

if (key < node->key)

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

else if (key > node->key)

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

return node;

}

int main()

{

struct node \*root = NULL;

root = insert(root, 'M');

insert(root, 'N');

insert(root, 'O');

insert(root, 'P');

insert(root, 'Q');

cout<<"\nPre-Order \n";

print\_preorder(root);

cout<<"\nIn-order"<<endl;

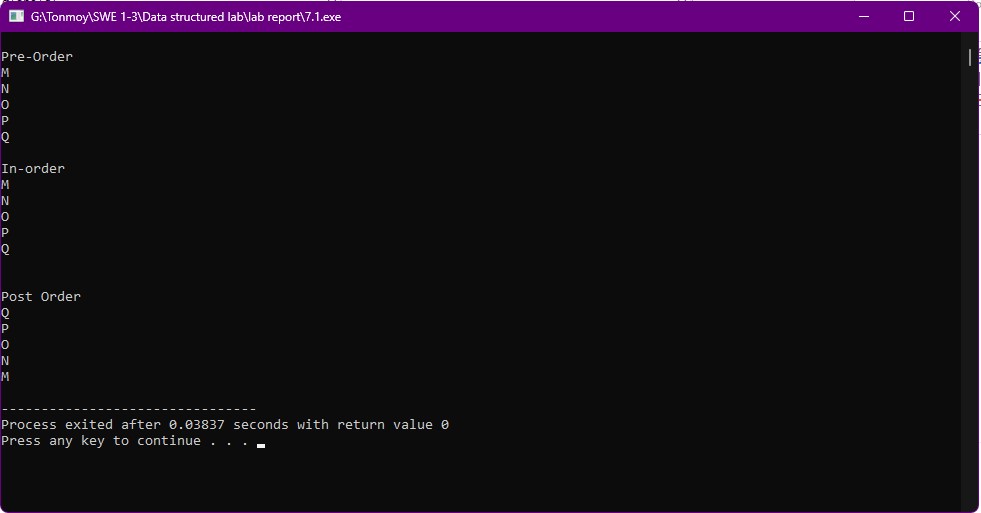
inorder(root);

cout<<"\n\n"<<"Post Order \n";

print\_postorder(root);

return 0;

}



9.2

#include<bits/stdc++.h>

using namespace std;

struct node

{

int key;

struct node \*left, \*right;

};

struct node \*newNode(int item)

{

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

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

void inorder(struct node \*root)

{

if (root != NULL)

{

inorder(root->left);

cout<<root->key<<endl;

inorder(root->right);

}

}

void print\_preorder(struct node \* root)

{

if (root)

{

cout<<root->key<<endl;

print\_preorder(root->left);

print\_preorder(root->right);

}

}

void print\_postorder(struct node \* root)

{

if (root)

{

print\_postorder(root->left);

print\_postorder(root->right);

cout<<root->key<<endl;

}

}

struct node\* insert(struct node\* node, int key)

{

if (node == NULL) return newNode(key);

if (key < node->key)

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

else if (key > node->key)

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

return node;

}

int main()

{

struct node \*root = NULL;

root = insert(root, 30);

insert(root, 15);

insert(root, 7);

insert(root, 22);

insert(root, 17);

insert(root, 27);

insert(root, 60);

insert(root, 45);

insert(root, 75);

cout<<"\nPre-Order \n";

print\_preorder(root);

cout<<"\nIn-order"<<endl;

inorder(root);

cout<<"\n\n"<<"Post Order \n";

print\_postorder(root);

return 0;

}

