Task 1

#include<iostream>

using namespace std;

const int tableSize = 10;

struct Node{

int data;

Node\* next;

};

class HashTable{

private:

Node\* table[tableSize];

int hashFun(int key){

return key%tableSize;

}

public:

HashTable(){

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

table[i]=NULL;

}

}

void insert(int key){

int index = hashFun(key);

Node\* newNode = new Node;

newNode->data= key;

newNode->next= NULL;

if(table[index]==NULL){

table[index]=newNode;

}else{

newNode->next = table[index];

table[index]=newNode;

}

cout<<"Data " <<key<<" inserted at "<<index<<endl;

}

void search(int key){

int index = hashFun(key);

Node\* current = table[index];

while(current!=NULL){

if(current->data == key){

cout<<"Data "<<key<<" Found at "<<index<<endl;

return;

}

current=current->next;

}

cout<<"Element Not Present!"<<endl;

}

void deletee(int key){

int index = hashFun(key);

Node\* current = table[index];

Node\* temp = NULL;

while(current!=NULL){

if(current->data = key){

if(temp == NULL){

table[index]=current->next;

} else{

temp->next = current->next;

}

delete current;

cout<<"Data " <<key <<" Deleted from index "<<index<<endl;

return;

}

temp = current;

current = current->next;

}

cout<<"Data not present "<<endl;

}

};

int main(){

HashTable hashtable;

hashtable.insert(20);

hashtable.insert(34);

hashtable.insert(45);

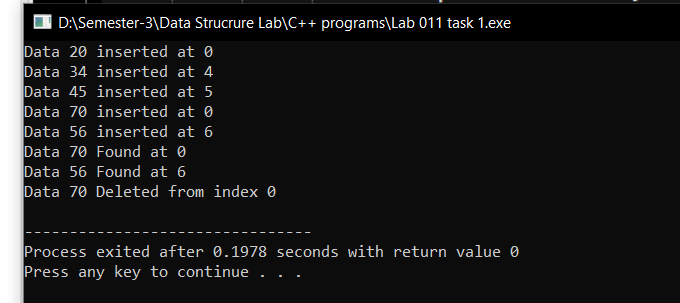
hashtable.insert(70);

hashtable.insert(56);

hashtable.search(70);

hashtable.search(56);

hashtable.deletee(70);

}

Task 2

#include<iostream>

using namespace std;

int solve(int arr[],int n,int sum){

int i =0 ;

int j=n-1;

int k =0;

int count =0;

while(i<j){

k=arr[i]+arr[j];

if(k==sum){

count++;

i++;

j--;

}else if(k>sum){

j--;

}else {

i++;

}

}

cout<<"Total pairs are "<<count<<endl;

}

int main() {

int arr[] = {1,2,3,4,5,6,7};

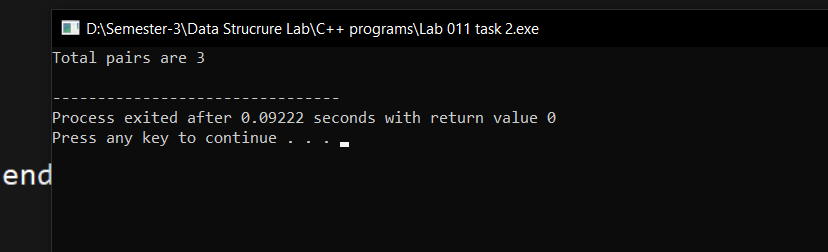
int size = sizeof(arr) / sizeof(arr[0]);

solve(arr,size,8);

// cout<<size;

return 0;

}



Task 3

#include<iostream>

using namespace std;

int solve(int arr[],int n){

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

int l = i+1;

int r = n-1;

while(l<r){

int sum = arr[i]+arr[l]+arr[r];

if(sum ==0){

return 1;

}else if (sum>0){

r--;

}else{

l++;

}

}

return 0;

}

}

int main(){

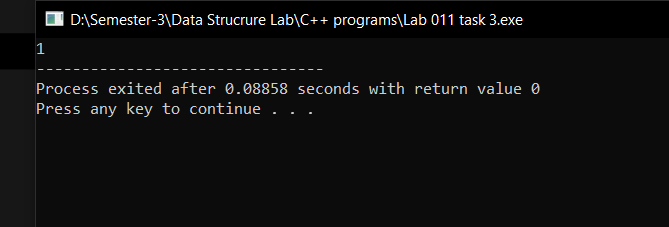
int arr[] = {0, -1, 2, -3, 1};

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

int result = solve(arr,size);

cout<<result;

return 0;

}

Task 4

#include<iostream>

using namespace std;

int partition(char arr[],int low,int high,char pivot){

int i =low;

char temp1,temp2;

for(int j=low;j<high;j++){

if(arr[j]<pivot){

temp1 = arr[i];

arr[i]=arr[j];

arr[j]=temp1;

i++;

}

else if(arr[j]==pivot){

temp1 = arr[j];

arr[j]=arr[high];

arr[high]=temp1;

j--;

}

}

temp2 =arr[i];

arr[i]=arr[high];

arr[high]=temp2;

return i;

}

void match(char nuts[], char bolts[], int low, int high){

if(low<high){

int pivot = partition(nuts,low,high,bolts[high]);

partition(bolts,low,high,nuts[pivot]);

match(nuts,bolts,low,pivot-1);

match(nuts,bolts,pivot+1,high);

}

}

void print(char arr[]){

for(int i=0;i<5;i++){

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

}

cout<<endl;

}

int main(){

char nuts[]={'@', '%', '$', '#', '^'};

char bolts[] ={'%', '@', '#', '$', '^'};

match(nuts,bolts,0,4);

print(nuts);

print(bolts);

return 0;

}