Question 1

#include <iostream>

using namespace std;

int Partition(int arr[],int s,int e){

int pivot=arr[e];

int pindex=s;

for(int i=s;i<e;i++){

if(arr[i]<=pivot){

int temp;

temp=arr[i];

arr[i]=arr[pindex];

arr[pindex]=temp;

pindex++;

}

}

int temp;

temp=arr[e];

arr[e]=arr[pindex];

arr[pindex]=temp;

return pindex;

}

void QuickSort(int arr[],int s,int e){

if(s<e){

int p=Partition(arr,s,e);

QuickSort(arr,s,p-1);

QuickSort(arr,p+1,e);

}

}

int main()

{

int arr [] {7,9,3,2,6};

cout<<"before sorting : ";

for(auto i:arr){

cout<<i<<" ";

}cout<<endl;

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

int s=5;

QuickSort(arr,0,s-1);

cout<<"After sorting : ";

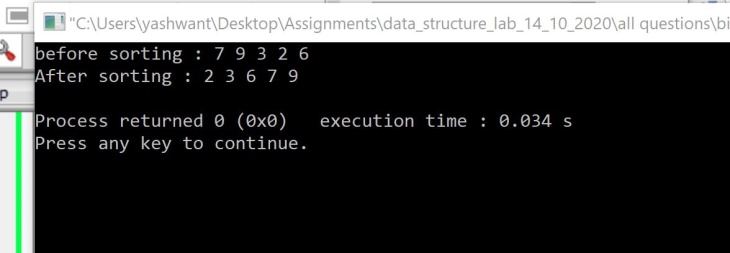
for(auto i:arr){

cout<<i<<" ";

}cout<<endl;

return 0;

}



2.Write a program to implement Quick Sort with random pivot selection.

#include <iostream>

#include <cstdlib>

using namespace std;

//void Swap(int &arr[i])

int Partition(int arr[],int s,int e){

int pivot=arr[e];

int pindex=s;

for(int i=s;i<e;i++){

if(arr[i]<=pivot){

swap(arr[i],arr[pindex]);

pindex+=1;

}

}

swap(arr[pindex],arr[e]);

return pindex;

}

int PartitionRand(int arr[],int s,int e){

// srand(time(NULL));

int random=s+rand()%(e-s);

swap(arr[random],arr[e]);

return Partition(arr,s,e);

}

void QuickSort(int arr[],int s,int e){

if(s<e){

int p=Partition(arr,s,e);

QuickSort(arr,s,p-1);

QuickSort(arr,p+1,e);

}

}

int main()

{

int arr [] {7,9,3,2,6};

cout<<"before sorting : ";

for(auto i:arr){

cout<<i<<" ";

}cout<<endl;

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

int s=5;

QuickSort(arr,0,s-1);

cout<<"After sorting : ";

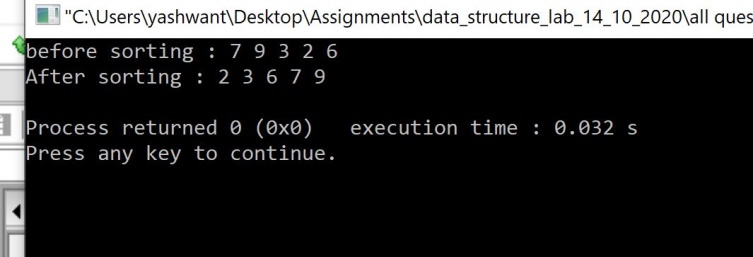
for(auto i:arr){

cout<<i<<" ";

}cout<<endl;

return 0;

}



3.Write a program to implement Quick Sort withmedianas pivot.

#include <iostream>

#include <cstdlib>

using namespace std;

//void Swap(int &arr[i])

int Partition(int arr[],int s,int e){

int pivot=arr[e];

int pindex=s;

for(int i=s;i<e;i++){

if(arr[i]<=pivot){

swap(arr[i],arr[pindex]);

pindex+=1;

}

}

swap(arr[pindex],arr[e]);

return pindex;

}

int PartitionRand(int arr[],int s,int e){

// srand(time(NULL));

int median;

if((e-s+1)%2){

median=s+(e-s)/2;

}

else{

median=median=s+(e-s)/2;

}

swap(arr[median],arr[e]);

return Partition(arr,s,e);

}

void QuickSort(int arr[],int s,int e){

if(s<e){

int p=Partition(arr,s,e);

QuickSort(arr,s,p-1);

QuickSort(arr,p+1,e);

}

}

int main()

{

int arr [] {7,9,3,2,6,8,-4,2,-33};

cout<<"before sorting : ";

for(auto i:arr){

cout<<i<<" ";

}cout<<endl;

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

QuickSort(arr,0,s-1);

cout<<"After sorting : ";

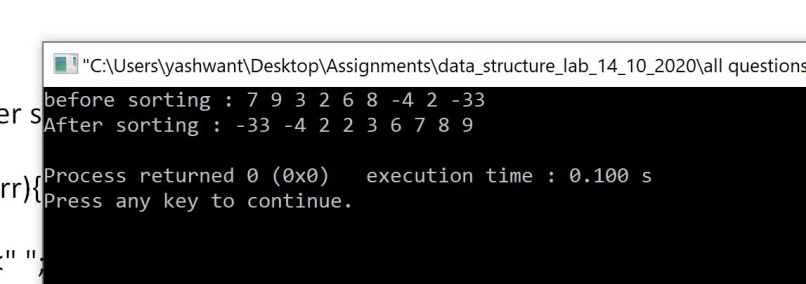
for(auto i:arr){

cout<<i<<" ";

}cout<<endl;

return 0;

}



4.Write a program to check the balanced parentheses of an expression.

#include <iostream>

#include <cstdlib>

#include <string>

#include <stack>

using namespace std;

bool check(string exp){

stack <char> stk;

for(int i=0;i<exp.length();i++){

if(!stk.size()){

return false;

}

if(exp[i]=='{'||exp[i]=='('||exp[i]=='['){

stk.push(exp[i]);

}

else if(exp[i]=='}'){

if(stk.top()!='{')

return false;

stk.pop();

}

else if(exp[i]==')'){

if(stk.top()!='(')

return false;

stk.pop();

}

else if(exp[i]==']'){

if(stk.top()!='[')

return false;

stk.pop();

}

}

if(stk.size()){

cout<<stk.top()<<endl;

return false;

}

return true;

}

int main()

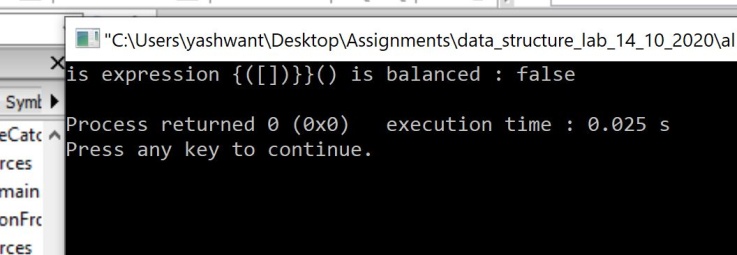
{

string expression="{([])}}()";

cout<<boolalpha;

cout<<"is expression is balanced : "<<check(expression)<<endl;

return 0;

}

5.Writea program fordecimal to binary conversion using stack.

#include <iostream>

#include <stack>

#include <string>

using namespace std;

string bn(int num){

stack <int> stk;

string b="";

int r;

while(num){

r=num%2;

stk.push(r);

num=num/2;

}

while (stk.size()){

b=b+to\_string(stk.top());

stk.pop();

}

return b;

}

int main(){

int decimalNumber=7883784;

cout<<"binary representation of "<<decimalNumber<<" in binary is "<<bn(decimalNumber)<<endl;

return 0;}

