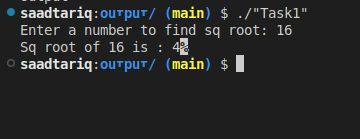
**Saad Tariq**

**22f8785**

**Assignment 1**

**Algorithms**

**Task 1**

****

#include<iostream>

using namespace std;

int sqRoot(int num);

int main()

{

int num=0;

cout<<"Enter a number to find sq root: ";

cin>>num;

cout<<"Sq root of "<<num<<" is : "<<sqRoot(num);

return 0;

}

int sqRoot(int num)

{

int low=0;

int high=num;

int mid;

while (low <= high)

{

mid = (low + high) / 2;

if (mid \* mid == num)

return mid;

if (mid \* mid > num)

high = mid - 1;

else

{

low = mid + 1;

mid = low - 1;

}

}

return mid;

}

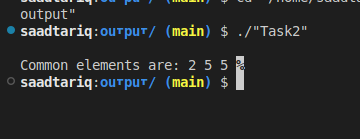
**Task 2**

#include <iostream>

using namespace std;

int\* findCommonElements(int\* list1,int sz1 , int\* list2, int sz2, int&count)

{

count=0;

int i = 0, j = 0;

int sz3=sz1<sz2?sz1:sz2;

int\* result=new int[sz3];

while (i < sz1 && j < sz2)

{

if (list1[i] == list2[j])

{

result[count]=list1[i];

count++;

i++;

j++;

} else if (list1[i] < list2[j])

{

i++;

} else

{

j++;

}

}

return result;

}

int main() {

int list1[] = {2, 5, 5, 5};

int list2[] = {2, 2, 3, 5, 5, 7};

int sz1 = sizeof(list1)/sizeof(list1[0]);

int sz2 = sizeof(list2)/sizeof(list2[0]);

int resSz=0;

int\* res = findCommonElements(list1,sz1, list2,sz2,resSz);

cout << "\nCommon elements are: ";

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

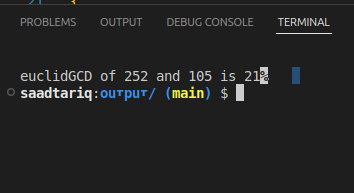
{

cout << res[i] << " ";

}

return 0;

}

**Task 3**

#include<iostream>

using namespace std;

int EuclidGcd(int num1, int num2);

int main()

{

system("clear");

// int a=14142;

// int b=31415;

int a=252;

int b=105;

int euclidGCD=EuclidGcd(a,b);

cout<<"\neuclidGCD of "<< a <<" and "<< b <<" is "<<euclidGCD;

return 0;

}

int EuclidGcd(int num1, int num2)

{

int a = num1>num2?num1:num2;

int b = num1>num2?num2:num1;

int r=-1;

while(r!=0)

{

r=a%b;

a=b;

b=r;

}

return a;

}

**Comparison:**

* Euclid’s algorithm required 10 steps.
* The consecutive integer checking algorithm could require up to 14142 steps.
* The ratio of steps is approximately 14142 / 10 = 1414.2.

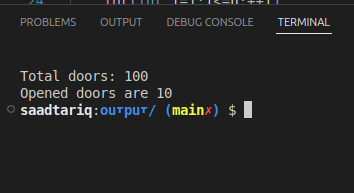
**Euclid's algorithm is approximately 1414 times faster than the consecutive integer checking algorithm in this case.**

**Task 4**

#include<iostream>

using namespace std;

void toggleLockers(bool\* doors, int n);

int main()

{

system("clear");

int n =100;

bool \* doors = new bool[n+1];

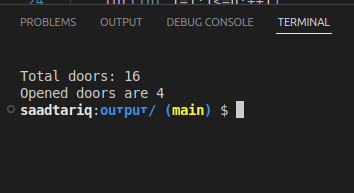
for(int i=1;i<=n;i++)

{

doors[i]=0;

}

toggleLockers(doors, n);



delete[] doors;

return 0;

}

void toggleLockers(bool\* doors, int n)

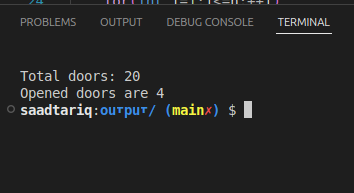
{

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

{

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

{

doors[j]=!doors[j];

}

}

string door="";

int openDoors=0;

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

{

if(doors[i]) openDoors++;

}

cout<<"\nTotal doors: "<<n;

cout<<"\nOpened doors are "<<openDoors<<"\n";

}

# Task 5

The peasant can safely transport all by following steps:

* he first takes the goat across the river. The wolf can’t eat the cabbage.
* He then takes the wolf across and brings the goat back. Peasant is present so goat not eat cabbage.
* Then he takes the cabbage across. The wolf can’t eat the cabbage.
* Finally he takes the goat across .

All are safely transported.

# Task 10

For every point P(x,y) in set

dist = compute the distance of P from origin

Add dist to an array distArr

Now

For i=0 to i<= length(distArr) - 1

check if distArr[i] is not equal to distArr[i+1]

print P[i] not lie on cicumference

else

Print P[i] lie on circumference

# Task 14

a. Show the stack after each operation of the following sequence that starts with the empty stack:

**push(a), push(b), pop, push(c), push(d), pop**

* a
* ba
* a
* ca
* dca
* ca

b. Show the queue after each operation of the following sequence that starts with the empty queue:

**enqueue(a), enqueue(b), dequeue, enqueue(c), enqueue(d), dequeue**

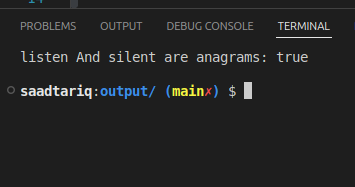
* a
* ab
* a
* ac
* acd
* ac

# Task 17

#include<iostream>

using namespace std;

bool IsAnagram(string word1, string word2);

int main()

{

system("clear");

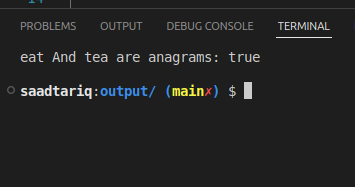
string w1="listen";

string w2="silent";

bool anagram = IsAnagram(w1,w2);

string res= anagram?"true":"false";

cout<<w1<<" And "<<w2<<" are anagrams: "<<res<<endl<<endl;

return 0;

}

bool IsAnagram(string word1, string word2)

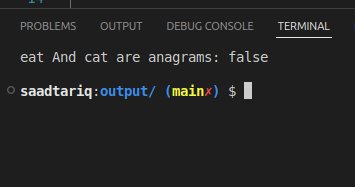
{

if(word1.length()!=word2.length())

{

return false;

}

int count=0;

int len=word1.length();

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

{

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

{

if(word1[i]==word2[j])

{

count++;

}

}

}

if(count!=len)

return false;

return true;

}