

**Programming Test**

This paper has been designed to test your programming abilities – it is meant to be a starting point for further discussion and provides you an opportunity to highlight your skills. You can take the test in the programming language of your choice. Your code doesn’t have to compile exactly but it should be reasonably close (pseudo-code is not OK).

If you are unsure about something, or make an assumption, write the assumption as part of your answer. Good luck ☺

**A-1**: Write a function that takes input as a String. It should return true if all characters in the string are

unique, false otherwise. You cannot use additional data structure. (a-z and 1-9. No Unicode/Special

character/Uppercase characters)

Example:

Input#1: raja

Output#1: false

Input#2: abc

Output#1: true

**//solution :(language C++)**

**#include <iostream>**

**#include<bits/stdc++.h>**

**using namespace std;**

**bool solution(string str){**

**sort(str.begin(),str.end());**

**for(int i=0;i<str.size()-1;i++){**

**if(str[i]==str[i+1])**

**return false;**

**}**

**return true;**

**}**

**int main()**

**{**

**string str;**

**cin>>str;**

**if(solution(str))**

**cout<<"true"<<endl;**

**else**

**cout<<"false";**

**return 0;**

**}**

**A-2**: Write a function that takes input as an integer number and prints the closest prime integer to that

number. The closest prime can be greater or smaller than the passed input integer. If there are equi-

distant prime-numbers, print both.

Example:

Input#1: 32

Output#1: 31

Input#2: 30

Output#2: 29 31

**Solution in c++:**

**#include <iostream>**

**#include<bits/stdc++.h>**

**using namespace std;**

**int primeleft(int n)**

**{**

**if (n & 1)**

**n -= 2;**

**else**

**n--;**

**int i, j;**

**for (i = n; i >= 2; i -= 2) {**

**if (i % 2 == 0)**

**continue;**

**for (j = 3; j <= sqrt(i); j += 2) {**

**if (i % j == 0)**

**break;**

**}**

**if (j > sqrt(i))**

**return i;**

**}**

**return 2;**

**}**

**int primeright(int n)**

**{**

**if(n%2==0)**

**n+=1;**

**int i, j;**

**bool prime=true;**

**for (i = n; i < INT\_MAX; i=i+2) {**

**prime=true;**

**for (j = 2; j\*j <=i; j ++) {**

**if (i % j == 0){**

**prime=false;**

**break;**

**}**

**}**

**if(prime)**

**return i;**

**}**

**return 0;**

**}**

**int main()**

**{**

**int num;**

**cin>>num;**

**int l=primeleft(num);**

**int r=primeright(num);**

**if(num-l==r-num){**

**cout<<l<<" "<<r<<endl;**

**}else if(num-l >r-num){**

**cout<<r<<endl;**

**}else**

**cout<<l<<endl;**

**return 0;**

**}**

**A-3:** Implement a method to perform basic string compression using the counts of repeated characters. E.g.; the string aabcccccaaa would become a2b1c5a3. If “compressed” string would not be smaller than the original string, your method should return the original string. Assume string has only lowercase letters (a-z).

For “aabcccccaaa” input, your method will return “a2b1c5a3” but for “abcd” input, your method will return “abcd”

The method would take input as the string and return the compressed string per above logic.

Solution in C++:

#include<bits/stdc++.h>

using namespace std;

string comp(string str){

vector<int>v(26,0);

for(int i=0;i<str.size();i++){

v[str[i]-'a']=v[str[i]-'a']+1;

}

string str1;

int j=0;

for(int i=0;i<26;i++){

if(v[i]!=0){

str1+=i+'a';

string st= to\_string(v[i]);

str1+=st;

}

}

if(str.size()==str1.size())

return str;

return str1;

}

int main()

{

string str;

cin>>str;

cout<<comp(str);

return 0;

}

**A-4:** Write a program that prints the numbers from 100 to 500.

● But for multiples of 7, print “Cool” instead of the number

● For multiples of 11 print “Dude” instead of the number.

● For numbers which are multiples of both 7 and 11 print “Cool Dude” instead of the number

//Solution in C++:

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int main()

{

for(int num=100;num<501;num++){

if(num%7==0&&num%11==0){

cout<<"Cool Dude"<<" ";

}else if(num%7==0){

cout<<"Cool"<<" ";

}else if(num%11){

cout<<"Dude"<<" ";

}else

cout<<num<<" ";

}

return 0;

}

**A-5**: **Variation of Fibonacci**

The Fibonacci sequence is constructed by adding the last two numbers of the sequence so far to get the next number in the sequence. The first and the second numbers of the sequence are defined as 0 and 1. We get:  
0, 1, 1, 2, 3, 5, 8, 13, 21…

Write a function which takes input as a number:

* If the given number is a Fibonacci number, print the number
* If the given number is NOT Fibonacci number, print the sum of all even Fibonacci numbers less than the given number.

int getFibOutput(int num) {

// TODO:

if(num==0 || num==1){

cout<<num;

return 0;

}

int v=1;

int w=0;

long long add=1;

while(v<num){

int temp=v+w;

if(temp<num)

add+=temp;

w=v;

v=temp;

}

if(v==num){

cout<<num<<endl;

}else{

cout<<add<<endl;

}

}

Example

(21 is a Fibonacci number)

Input: 21 Output: 21

(20 is NOT a Fibonacci number so, output is 10 (2+8))

Input: 20 Output: 10