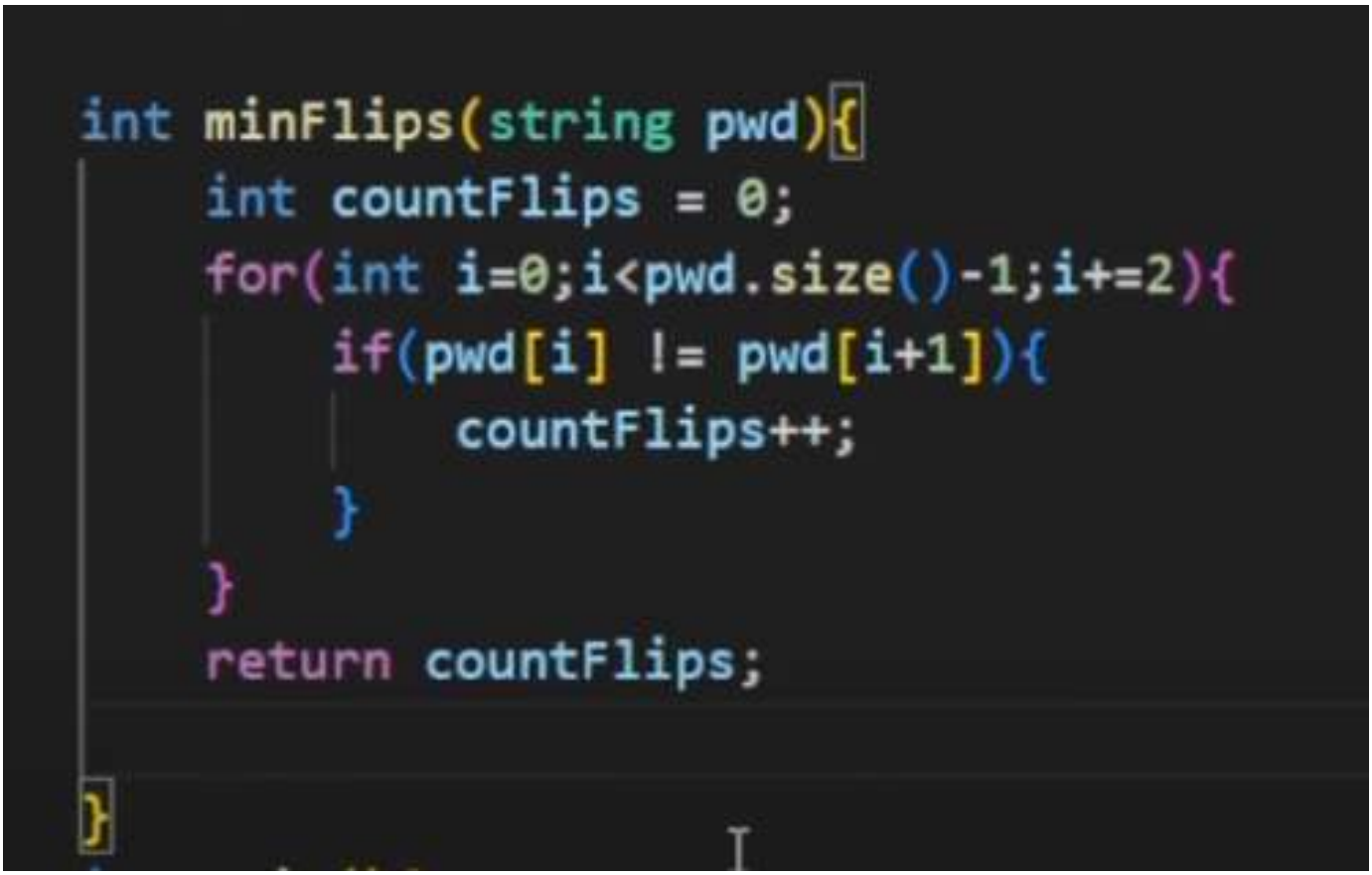

PDF SOLUTION OF ALL IBM ASSESSMENT QUESTIONS
IBM
IBM ALL CODING QUESTIONS SOLUTION
APNE WALE CODERS
ASSESSMENT

1. Question 1

A password string, *pwd*, consists of binary characters (0s and 1s). A cyber security expert is trying to determine the minimum number of changes required to make the password secure. To do so, it must be divided into substrings of non-overlapping, even length substrings. Each substring can only contain 1s or 0s, not a mix. This helps to ensure that the password is strong and less vulnerable to hacking attacks.

Find the minimum number of characters that must be flipped in the password string, i.e. changed from 0 to 1 or 1 to 0 to allow the string to be divided as described.


int minFlips(string pwd){
int countFlips = 0;
for(int i=0;i<pwd.size()-1;i+=2){
if(pwd[i] != pwd[i+1]){
countFlips++;
}
}
return countFlips;
}

2. Question 2

For each element of an array, a *counter* is set to *0*. The element is compared to each element to its left. If the element to the left is greater, the absolute difference is subtracted from the counter. If the element to the left is less, the absolute difference is added to the counter. For each element of the array, determine the value of the counter. These values should be stored in an array and returned.

Example
n = 3, the number of elements
arr = [2, 4, 3]

- For *arr[0] = 2*, *counter* starts at *0* and there are no elements to the left so *counter = 0*.
- For *arr[1] = 4*, *counter* starts at *0* and then increases by *| 4 - 2 | = 2* at the first and only comparison: *counter = 2*.
- Testing *arr[2] = 3*, first against *4*, *counter = 0 - |3 - 4| = -1*, and then against *2*, *counter = -1 + |3 - 2| = 0*.
- The answer array is *[0, 2, 0]*.

```cpp
vector<long> arrayChallenge(vector<long> &arr){
    vector<long> ans;

    int n = arr.size();
    ans.push_back(0);

    for(int i=1;i<n;i++){
        int sum = 0;
        for(int j=0;j<i;j++){
            int ele = arr[i] - arr[j];
            sum+=ele;
        }
        ans.push_back(sum);
    }
    return ans;
}
```

1. Question 1

An English lecture at HackerElementary School is aimed at teaching students the letters of the alphabet.

The students are provided with a string *word* that consists of lowercase English letters. In one move, they can choose any index *i*, and let the character at that index be *c*. Then, the first occurrence of *c* to the left of *i*, and the first occurrence of *c* to the right of *i* are deleted (Note: the operation can still be carried out even if either the left or right occurrence doesn't exist). For example, if *word* = *"adabacaea"*, and if index 4 is chosen (0-based), the first occurrence of 'a' to the left and right of index 4 (bold, indices 2 and 6) are deleted leaving *word* = *"adbacea"*.

Find the minimum number of moves the students need to perform in order to get a word of minimal length.

Example
Consider *word* = *"baabacaa"*.

The following moves are optimal.

1. Choose index 0, *"baabacaa"*, then *word* = *"baaacaa"*. Delete the *b* to its right at index 3. There is no ***b*** **to its left so the operation is finished.**
2. Now, choose index 2, *"baaacaa"*, then *word* = *"bacaa"*.
3. Now, choose index 3, *"bacaa"*, then *word* = *"bca"*.

The word cannot be reduced further. The answer is 3.

```
int findMinimumMoves(string &word){

    int moves = 0;
    unordered_map<char,int> cmp;
    for(int i=0;i<word.size();i++){
     cmp[word[i]]++;
    }

    for(auto it:cmp){
     if(it.second > 1){
         moves += (it.second)/2;
     }
    }
    return moves;
}
```

1. Question 1

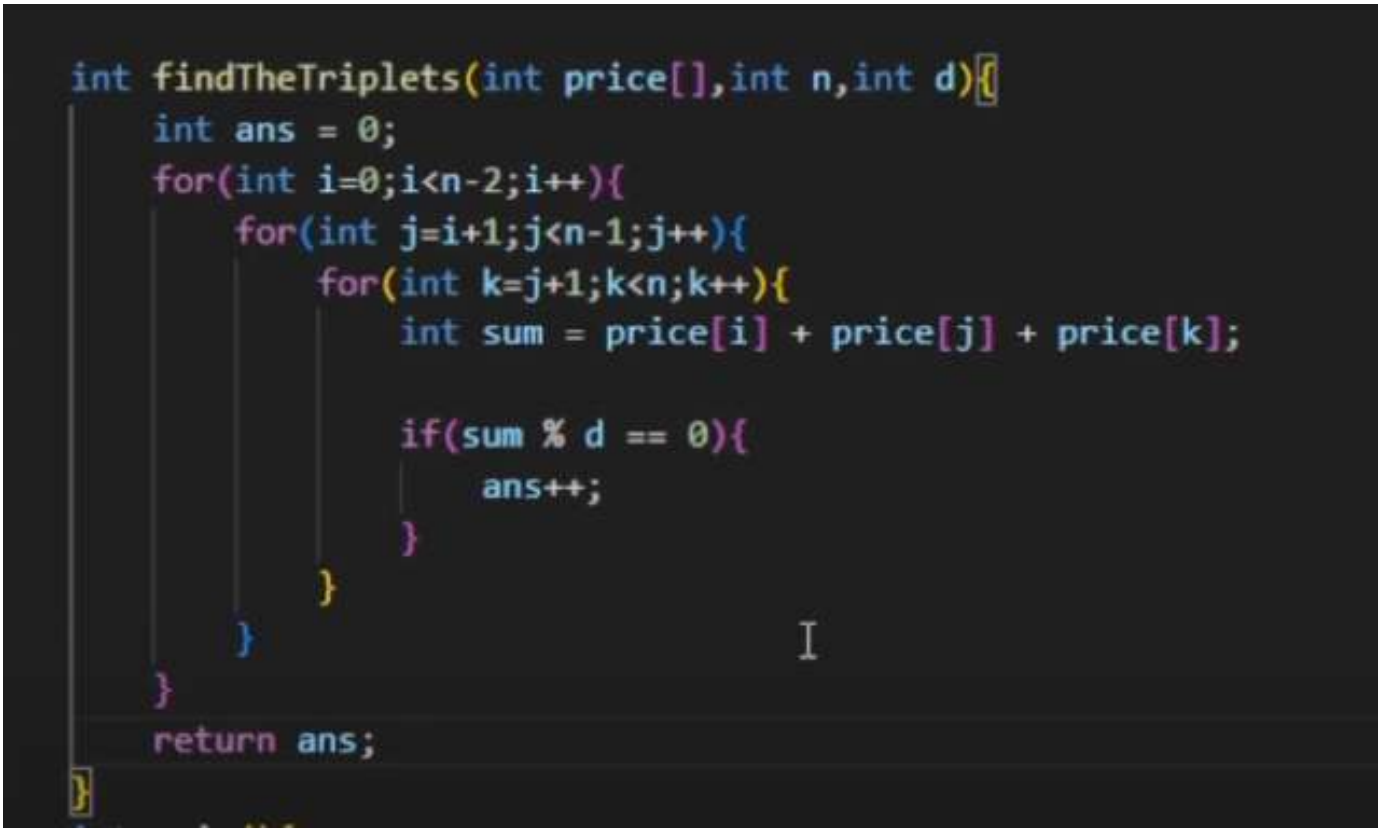
The cost of a stock on each day is given in an array, *arr*. An investor wants to buy the stocks in triplets such that the sum of the cost for three days is divisible by *d*.
The goal is to find the number of distinct triplets *(i, j, k)* such that $i < j < k$ and the sum *(a[i]+a[j]+a[k])* is divisible by *d*.

Example

Let *arr, prices of stock = [3, 3, 4, 7, 8]* and $d = 5$. Following are the triplets whose sum is divisible by *d (1-based indexing)*:

- Triplet with indices - (1, 2, 3), sum = 3+3+4 which is equal to 10
- Triplet with indices - (1, 3, 5), sum = 3+4+8 which is equal to 15
- Triplet with indices - (2, 3, 4), sum = 3+4+8 which is equal to 15

Hence the answer is 3.



```
int findTheTriplets(int price[],int n,int d){
    int ans = 0;
    for(int i=0;i<n-2;i++){
        for(int j=i+1;j<n-1;j++){
            for(int k=j+1;k<n;k++){
                int sum = price[i] + price[j] + price[k];

                if(sum % d == 0){
                    ans++;
                }
            }
        }
    }
    return ans;
}
```

1. FizzBuzz

Given a number *n*, for each integer *i* in the range from *1* to *n* inclusive, print one value per line as follows:

- If *i* is a multiple of both *3* and *5*, print *FizzBuzz*.
- If *i* is a multiple of *3* (but not *5*), print *Fizz*.
- If *i* is a multiple of *5* (but not *3*), print *Buzz*.
- If *i* is not a multiple of *3* or *5*, print the value of *i*.

Function Description

Complete the function *fizzBuzz* in the editor below.

fizzBuzz has the following parameter(s):

int n: upper limit of values to test (inclusive)

Returns: NONE

Prints:

The function must print the appropriate response for each value *i* in the set *{1, 2 ... n}* in ascending order, each on a separate line.

```
void fizzBuss(int n){
    for(int i=1;i<=n;i++){
        if(i % 3 == 0 && i % 5 == 0){
            cout<<"Fizzbuzz"<<endl;
        }
        else if(i % 3 == 0 && i % 5 != 0){
            cout<<"Fizz"<<endl;
        }
        else if(i % 3 != 0 && i % 5 == 0){
            cout<<"Buzz"<<endl;
        }
        else{
            cout<<i<<endl;
        }
    }
}
```

In the assembly line, the factory assembles three parts 'a' 'b' 'c' of a triangle toy. A valid toy is one where the two shorter sides added together are greater in length than the longest side.

There are two forms of valid triangles to identify.

- If 2 parts are of equal length, the form is 'Isosceles'
- If all 3 parts are of equal length, the form is 'Equilateral'

If a toy is not valid or is not one of those 2 forms, it is 'None of these'.

Example

triangleToy={'2 2 1', '3 3 3', '3 4 5', '1 1 3'}

```
vector<string> triangleType(const vector<string>& triangleToy) {
   vector<string> ans;

   for(const string& sides :triangleToy){
     stringstream ss(sides);
     int side1,side2,side3;

     ss>> side1 >> side2 >> side3;

    if((side1 + side2 > side3) && (side2 + side3 > side1) && (side1 + side3 > side2)){
        if(side1 == side2 && side2 == side3){
            ans.push_back("Equilateral");
        }else if(side1 == side2 || side2 == side3 || side3 == side1){
            ans.push_back("Isosceles");
        }
        else{
            ans.push_back("None of these");
        }
    }
     else{
            ans.push_back("None of these");
    }
   }
   return ans;
}
```

1. Question 1

Given *n* request ids as an array of strings, *requests*, and an integer *k*, after all requests are received, find the *k* most recent requests. Report the answer in order of most recent to least recent.

Example:

Suppose *n* = 5, *requests* = ["item1", "item2", "item3", "item1", "item3"], and *k* = 3.

Starting from the right and moving left, collecting distinct *requests*, there is "item3" followed by "item1". Skip the second instance of "item3" and find "item2". The answer is ["item3", "item1", "item2"].

Function Description

Complete the function *getLatestKRequests* in the editor below.

getLatestKRequests takes the following arguments:

str requests[n]: the request ids

int k: the number of requests to report

```cpp
vector<string> getLatestKRequest(const vector<string>& requests, int k) {
    vector<string> ans;
    unordered_map<string,bool> check;
    int cnt = 0;

    for(int i=requests.size()-1;i>=0;i--){
        if(cnt == k){
            return ans;
        }
        if(check[requests[i]] != true){
            cnt++;
            check[requests[i]] = true;
            ans.push_back(requests[i]);
        }
    }

    return ans;

}
```

There is a shop with old-style cash registers. Rather than scanning items and pulling the price from a database, the price of each item is typed in manually. This method sometimes leads to errors. Given a list of items and their correct prices, compare the prices to those entered when each item was sold. Determine the number of errors in selling prices.

Example

products = ['eggs', 'milk', 'cheese']
productPrices = [2.89, 3.29, 5.79]
productSold = ['eggs', 'eggs', 'cheese', 'milk']
soldPrice = [2.89, 2.99, 5.97, 3.29].

```
          Price
Product   Actual  Expected  Error
eggs      2.89    2.89
eggs      2.99    2.89      1
cheese    5.97    5.79      1
milk      3.29    3.29
```

The second sale of eggs has a wrong price, as does the sale of cheese. There are *2* errors in pricing.

Function Description

Complete the function *priceCheck* in the editor below.

```
int priceCheck(vector<string> products, vector<float> productPrices, vector<string>
productSold, vector<float> soldPrice) {
    unordered_map<string,float> priceMap;

    for(int i=0;i<products.size();i++){
        priceMap[products[i]] = productPrices[i];
    }

    int errorCount = 0;
    for(int i=0;i<productSold.size();i++){
        if(priceMap.find(productSold[i]) != priceMap.end()){
            if(priceMap[productSold[i]] != soldPrice[i]){
                errorCount++;
            }
        }
    }
    return errorCount;
}
```


APNE
WALE
CODERS
APNE WALE CODERS
BEST CODING CHANNEL
SUBSCRIBE FOR MORE