Algortihms-Lab3

Moisés Bernardo Suárez Gamez

September 2019

1 Problem A-Integer Prefix

1.1 Code

```
#include <bits/stdc++.h>
#include <string.h>

using namespace std;

int main(){
    string s;
    getline(cin, s);
    char c[s.size()];
    int val;

    for(int i = 0; i <= s.size(); i++){
        val = i;
        if ((int)s[i] < 48 || (int)s[i] > 57){
        break;
        }
    }

    if(val == 0){
        cout << "-1";
    }else{
        cout << s.substr(0,val);
    }
}
```

1.2 Description

The idea is really simple, we go through the whole string character by character til we find a character whose value doesn't correspond to a number in the ASCII code.If the number of the evaluated character doesn't correspond to a numerical value in the ASCII code then we save the position where this character was found(val) and we leave the loop.
Finally, if the value of our variable "val" is equal to 0, it means that no suffix (a number) was found in the string, otherwise we print the corresponding result, which is the prefix with all the numbers found.

2 Problem B-Boring Non-Palindrome

2.1 Code

```
#include <iostream>
#include <algorithm>

using namespace std;

bool palindrome(string s){
    if(s == string(s.rbegin(), s.rend())){
        return true;
    }else{
        return false;
    }
}
int main() {
    string bnp;
    getline(cin,bnp);
    string normal = bnp;
    string rev = "";
    reverse(bnp.begin(),bnp.end());

    for(int i = 0 ; i < normal.size(); i++){
        if(palindrome(normal.substr(i,normal.size()))){
            normal += rev;
            break;
        }else{
            rev = normal.substr(i,1)+rev;
        }
    }
   cout << normal;
}
```

2.2 Description

Ok first at all we have a function called palindrome which determines if some string it's a palindrome or not.

Our goal is to find the longest suffix that is palindrome, we'll go through the whole string verifying if each suffix is a palindrome, if the suffix found in the i-th position is not a palindrome then we keep the i-th character and concatenate it in a new empty string, so when a suffix is found that is palindrome the necessary characters will be added to convert the entire string to a palindrome (note that for any string a palindrome will be found, since any character is considered a palindrome string).

3 Problem C-Informants

3.1 Code

```
#include <iostream>
#include <cmath>
using namespace std;
int N;
int saysCorrect[25], saysWrong[25];

int GetNumBits(int i){
    int count = 0;
    while (i){
        if (i & 1)
        ++count;
        i >>= 1;
    }
    return count;
}

int main(){
    int a;
    while (cin >> N >> a, N)    {
        for (int i = 0; i < N; ++i)
            saysCorrect[i] = saysWrong[i] = 0;

        while (a--){
            int x, y;
            cin >> x >> y;
            --x;
            if (y > 0)
                saysCorrect[x] |= (1 << y - 1);
            else
                saysWrong[x] |= (1 << -y - 1);
        }

        const int Highest = (1 << N) - 1;
        int maxPositive = 0;
        for (int i = 1; i <= Highest; ++i)        {
            int off = (~i) & Highest;
            int numBits = GetNumBits(i);
            if (numBits < maxPositive)
                continue;
            bool possible = true;

            for (int j = 0; j < N; ++j){
                if ((i & (1 << j))
                    && ((off & saysWrong[j]) != saysWrong[j]
```

```
                    || (i & saysCorrect[j]) != saysCorrect[j])){
                        possible = false;
                        break;
                    }
                }
                if (possible){
                    maxPositive = max(maxPositive, GetNumBits(i));
                }
            }
            cout << maxPositive << '\n';
        }
}
```

3.2 Description

It is difficult to explain, but the idea is to keep in different arrays the number of people who tell the truth and the number of people who tell the lie, as we'll know due to the problem, the veracity of a person depends on the word of the others, the idea with these two arrangements is to store the values in the respective arrangement by adding a certain degree of distrust or trust for each of the people, which are those that correspond to each of the positions in the two arrays.

4 Problem D-Burger Time?

4.1 Code

```cpp
#include <iostream>
#include <string>
#include <algorithm>

using namespace std;

int main(){
    int L;
    while (cin >> L, L != 0){
        string S;
        cin >> S;
        int minDistance = L;
        int lastR = -L, lastD = -L;
        for (int i = 0; i < L; ++i){
            if (S[i] == 'Z'){
                minDistance = 0;
                break;
            }else if (S[i] == 'R'){
                minDistance = min(minDistance, i - lastD);
                lastR = i;
            }else if (S[i] == 'D'){
                minDistance = min(minDistance, i - lastR);
                lastD = i;
            }
        }
        cout << minDistance << endl;
    }
}
```

4.2 Description

We'll go across the road, first we cover the trivial case, when a Z is found in the string, in this case, the minimum distance corresponds to 0 and it wouldn't be necessary to continue traveling the road.

Otherwise we calculate the minimum distance if an R or a D is found, where this minimum distance is relative to each one, that is, if we find an R first, we find the minimum distance of this relative to a D, starting with the length of the road, which is the maximum distance, and in the same way for the D's, having a minimum distance referring to the R's found.