AtCoder problems

Aheer Srabon

1 Atcoder Beginner Contest 347

1.1 Task A - Divisible

Problem Statement: You are given positive integers N and K, and a sequence of length N. $A = (A_1, A_2, ..., A_N)$. Extract all elements of A that are multiples of K, divide them by K, and print the quotients.

Solution:

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

int main (int argc, char *argv[]) {
        int n, k, a[110];
        std::cin >> n >> k;

        for (int i = 0; i < n; i++) {
                std::cin >> a[i];
        }
        std::sort(a, a+n);

        for (int i = 0; i < n; i++) {
                if (a[i]%k == 0) {
                        std::cout << a[i]/k << ' ';
                }
        }

        return 0;
}
```

1.2 Task B - Substring

Problem Statement: You are given a string S consisting of lowercase English letters. How many different non-empty substrings does S have? A substring is a contiguous subsequence. For example, xxx is a substring of yxxxy but not of xxyxx.

Solution:

```
#include <iostream>
#include <set>

int main (int argc, char *argv[]) {
        std::string s;
        std::cin >> s;
        std::set<std::string> t;

        for (int i = 0; i < s.size(); i++) {
                for (int j = 1; j <= s.size() - i; j++) {
                        t.insert(s.substr(i, j));
                }
        }
        std::cout<< t.size() << std::endl;

        return 0;
}
```

1.3 Task C - Ideal Holidays

Problem statement: In the kingdom of AtCoder, a week consists of $A + B$ days, with the first through A-th days being holidays and the $(A + 1)$-th through $(A + B)$-th being weekdays. Takahashi has N plans, and the i-th plan is scheduled D_i days later.
He has forgotten what day of the week it is today. Determine if it is possible for all of his N plans to be scheduled on holidays.

Constraints:

- $1 \leq N \leq 2 \times 10^5$
- $1 \leq A, B \leq \times 10^9$
- $1 \leq D_1 < D_2 < \cdots < D_N \leq \times 10^9$

Solution:

```
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

int main() {
        long long N, A, B;
        cin >> N >> A >> B;
        vector<long long> D(2*N);
        for (int i = 0; i < N; i++) {
                cin >> D[i];
                D[i] %= A+B;
                D[N+i] = D[i] + A + B;
```

```
    }

    sort(D.begin(), D.end());
    bool ans = false;

    for (int i = 0; i < N; i++) {
        int c = lower_bound(D.begin(), D.end(), D[i] + A) - D.begin();
        ans |= c == N + i;
    }

    cout << (ans ? ”Yes\n” : ”No\n”);

    return 0;
}
```

The method lower_bound(first, last, val) in C++ is used to return an iterator pointing to the first element in the range [first, last) which has value not less than val. So, the function returns an iterator pointing to the next smallest number just greater than or equal to that number. If there are multiple values that are equal to val, lower_bound() returns the iterator of the first such value. It operates on *sorted* vector. The first two arguments are iterators. The return value of lower_bound() is an iterator to the

2 Atcoder Regular Contest 175

2.1 Task A - Spoon Taking Problem

Solution:

```
// Solution by Hlcoder

#include<iostream>
#include<set>
#include<algorithm>
#include<vector>
#include<string>
#include<set>
#include<map>
#include<stack>
#include<numeric>
#include<queue>
#include<cassert>
#include<cmath>
#include<deque>
using namespace std;
typedef long long ll;
const ll INF=1LL<<60;
typedef pair<int,int> P;
typedef pair<ll,P> PP;
```

```cpp
const int mod = 998244353;
struct mint {
  long long x; // typedef long long long long;
  mint(long long x=0):x((x%mod+mod)%mod){}
  mint operator-() const { return mint(-x);}
  mint& operator+=(const mint a) {
    if ((x += a.x) >= mod) x -= mod;
    return *this;
  }
  mint& operator-=(const mint a) {
    if ((x += mod-a.x) >= mod) x -= mod;
    return *this;
  }
  mint& operator*=(const mint a) { (x *= a.x) %= mod; return *this;}
  mint operator+(const mint a) const { return mint(*this) += a;}
  mint operator-(const mint a) const { return mint(*this) -= a;}
  mint operator*(const mint a) const { return mint(*this) *= a;}
  bool operator==(const mint a) const {return a.x==x;}
  mint pow(long long t) const {
    if (!t) return 1;
    mint a = pow(t>>1);
    a *= a;
    if (t&1) a *= *this;
    return a;
  }

  // for prime mod
  mint inv() const { return pow(mod-2);}
  mint& operator/=(const mint a) { return *this *= a.inv();}
  mint operator/(const mint a) const { return mint(*this) /= a;}
};

std::istream& operator>>(std::istream& is, mint& a) { return is >> a.x;}
std::ostream& operator<<(std::ostream& os, const mint& a) { return os << a.x;}



struct combination {
  std::vector<mint> fact, ifact;
  combination(int n):fact(n+1),ifact(n+1) {
    assert(n < mod);
    fact[0] = 1;
    for (int i = 1; i <= n; ++i) fact[i] = fact[i-1]*i;
    ifact[n] = fact[n].inv();
    for (int i = n; i >= 1; --i) ifact[i-1] = ifact[i]*i;
  }
  mint operator()(int n, int k) {
    if (k < 0 || k > n) return 0;
    return fact[n]*ifact[k]*ifact[n-k];
  }
};

int main(){
    int N;
    cin>>N;
    vector<int> p(N);
    for(int i=0;i<N;i++){
        cin>>p[i];
        p[i]--;
    }
    string S;
    cin>>S;

    mint ans=0;

    {
        mint ans_left=1;
        vector<bool> used_left(N,1);

        for(int i=0;i<N;i++){
            int pos=p[i];
            int rhs=(pos+1+N)%N;
            if(used_left[rhs]){
                if(S[pos]=='R')ans_left*=0;
            }else{
                if(S[pos]=='?')ans_left*=2;
            }

            used_left[pos]=0;

        }
        //cout<<"ans_left="<<ans_left<<endl;

        ans+=ans_left;

    }

    {
        mint ans_right=1;
        vector<bool> used_right(N,1);

        for(int i=0;i<N;i++){
            int pos=p[i];
            int lhs=(pos-1+N)%N;
            if(used_right[lhs]){
                if(S[pos]=='L')ans_right*=0;
            }else{
                if(S[pos]=='?')ans_right*=2;
            }

            used_right[pos]=0;

        }
        //cout<<”ans_right=”<<ans_right<<endl;

        ans+=ans_right;

    }

    cout<<ans<<endl;
}
```