Задача B2. ЗАПЛАЩАНЕ

Както е известно, поради проблеми със софтуерната система, обслужваща клиентите на електроразпределение, около Нова година опашките за заплащане на потреблението на електричество във Велико Търново нараснаха неимоверно. Хората се притесняваха, че поради неплатени сметки могат да останат без електрозахранване точно за празниците. Накрая въпросът беше решен, но пък възникна друг проблем – касите не можеха да се справят с бързащите да платят сметките си клиенти. Опашките нарастваха постоянно, започнаха скандали, на места се стигна даже до бой. За разрешаване на проблема с чакащите да платят клиенти, на всеки клиент още при пристигането му се дава входящ номер. За да се спазва предимството на майките с малки деца и възрастните хора, входящият номер се състои от главна латинска буква и поредния номер на клиента. Буквата може да бъде: *M* – майка с малко дете или бременна жена; *O* – възрастен човек; *R* – обикновен гражданин.

Клиентите се обработват по реда на пристигането си, като се дава предимство на възрастните хора пред обикновените граждани и на майките с деца – пред обикновените граждани и пред възрастните хора. Известно е, че докато касата обработи един клиент, на опашката се натрупват още 3 нови клиента. При отваряне на гишето за плащане има вече опашка от 10 чакащи клиента. На ден се обработват по не повече от 1 000 клиента.

Напишете програма **PAYMENT**, която намира поредния номер, под който е обслужен даден клиент.

От първия ред на стандартния вход се въвежда номера на пристигане на даден клиент. От следващия ред се въвеждат входящите номера на клиентите в реда на постъпването им. Номерата са разделени с по един интервал. На стандартния изход се извежда номера, под който е обслужен клиента.

ПРИМЕР

Вход

```
12
R1 R2 O3 R4 R5 R6 O7 R8 R9 O10 M11 M12 R13 M14 O15
```

Изход

```
3
```

Решение:

Задачата е типичен пример за използване на структура Опашка. В случая трябва да се генерират три опашки – по една за всеки тип клиент.

Данните постъпват последователно от входния файл и всеки номер се поставя в съответната опашка в зависимост от буквата пред него. В началото се прочитат последователно 10 поредни номера, а след това се изважда един номер от някоя от опашките и се четат по 3 нови номера.

Изваждането на номер от опашка става като последователно се сканират трите опашки според зададения в задачата приоритет. В дадена опашка се търси номер, само ако опашките с по-висок приоритет са празни. Процесът приключва при намиране на търсения елемент.

```
#include<iostream>
#include<cstdlib>
#include<queue>

using namespace std;

void Place(char num[]);
//Postavia poreden nomer v syotvetnata opashka

queue<int> M, O, R;
```

```
int main(){
     bool b;

     char num[7];
     int client, current, cnt=0;

     cin>>client;
     for (int i=0; i<10; i++){
          cin>>num;
          Place(num);
     }

     do {
          b = 0;
       //Kontrolira cikyla.
        //Ima stojnost 1, ako tyrseniat element ne e nameren
       //i opashkite ne sa prazi
        //i stojnost 1 - v protiven sluchaj -
        //nameren e element ili opashkite sa prazni
          if (!M.empty()) {current = M.front(); M.pop(); cnt++;
b=1;}
          else
               if (!O.empty()){current = O.front(); O.pop();
cnt++; b=1;}
               else
                    if (!R.empty()){
                    current = R.front(); R.pop(); cnt++; b=1;}

          if (b)
           if(current==client){
              cout<<cnt<<endl;
              b=0;
           }
           else
                 for (int i=0; i<3; i++){
                      cin>>num;
                      if (!cin.eof())
                      Place(num);
            }
     }while(b);

     return 0;
}

void Place(char num[]){
     char letter;
     letter = num[0];
     switch (letter){
          case 'M': M.push(atoi(num+1));
                         break;
          case 'O': O.push(atoi(num+1));
                         break;
          case 'R': R.push(atoi(num+1));
                         break;
     }
}
```