1.

Vremensko ograničenje	Memorijsko ograničenje	ulaz	izlaz
0,1 s	64 MB	standardni ulaz	standardni izlaz

Banka želi da upozori kupce na sumnjive aktivnosti na njihovom računu. Prilikom svake transakcije posmatra se središnja vrednost m (medijana) prethodnih d transakcija i ako je tekuća transakcija veća ili jednaka od dvostruke vrednosti m, izdaje se upozorenje. Središnja vrednost niza se izračunava tako što se pronađe središnji element u nizu sortiranom po veličini (ako je broj elemenata paran, onda se uzima aritmetička sredina između dva elementa oko sredine niza). Napiši program koji na osnovu spiska transakcija određuje broj upozorenja koja će banka se poslati.

Ulaz

Sa standardnog ulaza se unosi ceo broj n ($5 \leq n \leq 10^5$) koji određuje ukupan broj transakcija, zatim broj d ($1 \leq d \leq n$), a zatim n vrednosti transakcija (prirodni brojevi između 1 i 200). Svi brojevi se nalaze u posebnim linijama.

Izlaz

Na standardni izlaz ispisati jedan prirodan broj koji predstavlja broj sumnjivih transakcija.

Primer

Ulaz

```
8
3
2
5
3
4
3
6
2
9
```

Izlaz

```
2
```

Sumnjive transakcije su 6 (ona je jednaka dvostrukom iznosu medijane niza 3, 4, 3 koji je jednak 3) i 9 (ona je veća od dvostrukog iznosa medijane niza 3, 6, 2 koji je jednak 3).

REŠENJE

Ovo je jedan od zadataka u kojem se određuju statistike segmenata niza fiksirane dužine **(kaže se da ti segmenti predstavljaju klizajući prozor, tj. sliding window).**

U ovim situacijama najčešće nije potrebno čuvati istovremeno u memoriji sve članove serije (niza brojeva) koja se učitava sa standardnog ulaza, već je dovoljno elemente tekućeg segmenta (prozora) čuvati u FIFO redu (najlakši način za to je da upotrebimo bibliotečki tip `queue` tj. `Queue`).

Na početku programa popunjavamo red sa prvih d elemenata (iznosa prvih ddtransakcija) dodajući ih na kraj reda, a zatim u svakom narednom koraku pre dodavanja novog elementa na kraj, uklanjamo jedan element iz reda sa početka.

Jedna od specifičnosti teksta zadatka je to što su iznosi transakcija celi brojevi od 1 do 200. To sugeriše da će se u dužem prozoru naći veliki broj transakcija sa istim iznosima i da je pogodna reprezentacija prozora zapravo asocijativni niz u kojem se broje pojavljivanja svake transakcije.
Ažuriranje asocijativnog niza možemo vršiti inkrementalno, tako što svaki put kada dodamo iznos u prozor povećamo njegov broj pojavljivanja, a kada ga izbacimo iz prozora smanjimo njegov broj pojavljivanja.

Da bismo izbegli rad sa realnim brojevima, umesto medijane, možemo izračunavati njenu dvostruku vrednosti i nju porediti sa narednim iznosom. Složenost ovog koraka određena je izračunavanjem parcijalnih suma, a to je prilično brzo, pošto je niz broja pojavljivanja iznosa dužine najviše 200 (složenost je konstanta, doduše sa velikim konstantnim faktorom).

```
#include <iostream>

#include <deque>
#include <vector>
#include <algorithm>

using namespace std;


// vraca dvostruki iznos medijane elemenata datog reda
int dvostruka_medijana(const deque<int>& q) {
  // prebacujemo elemente reda u vektor (da bismo mogli da im menjamo
redosled)
  vector<int> q1(begin(q), end(q));

  // odredjujemo medijanu elemenata vektora
  if (q1.size() % 2 != 0) {
    // broj elemenata je neparan pa je medijana odredjena samo
    // sredisnjim elementom
    nth_element(begin(q1), next(begin(q1), q1.size() / 2), end(q1));
    return 2 * q1[q1.size() / 2];
  } else {
    // broj elemenata je neparan pa je medijana odredjena pomocu dva
    // elementa oko sredisnjeg
    nth_element(begin(q1), next(begin(q1), q1.size() / 2 - 1), end(q1));
    int a = q1[q1.size() / 2 - 1];
    nth_element(begin(q1), next(begin(q1), q1.size() / 2), end(q1));
    int b = q1[q1.size() / 2];
    return a + b;
  }
```

```
}

int main() {
  ios_base::sync_with_stdio(false);

  // ukupan broj transakcija
  int n;
  cin >> n;
  // sirina "prozora" koji sadrzi prethodne transakcije
  int d;
  cin >> d;
  // trenutni prozor (prethodnih d transakcija)
  deque<int> iznosi;

  // ucitavamo prvih d iznosa transakcija
  for (int i = 0; i < d; i++) {
    int iznos;
    cin >> iznos;
    iznosi.push_back(iznos);
  }

  // ukupan broj sumnjivih transakcija
  int broj_sumnjivih = 0;
  for (int i = d; i < n; i++) {
    // citamo novi iznos
    int iznos;
    cin >> iznos;

    // proveravamo da li je on sumnjiv
    if (iznos >= dvostruka_medijana(iznosi))
      broj_sumnjivih++;

    // uklanjamo stari iznos iz prozora
    iznosi.pop_front();
    // dodajemo novi iznos u prozor
    iznosi.push_back(iznos);
  }

  // ispisujemo konacan broj sumnjivih transakcija
  cout << broj_sumnjivih << endl;

  return 0;
}
```

2.

Vremensko ograničenje	Memorijsko ograničenje	ulaz	izlaz
1 s	32 MB	standardni ulaz	standardni izlaz

Spam poruke su sve prisutnije u našim životima. Jedna od čestih metoda kako se od njih možemo zaštititi je da se prilikom objavljivanja e-mail adrese na webu ili negde drugde one malo promene kako ih programi za automatsko traženje e-mail adresa ne bi uočili.
Za ovu priliku važe sledeća pravila za e-mail adrese:

1.) e-mail adresa je niz znakova koji se sastoji od malih slova engleske abecede ('a'...'z'), znakova '.' (tačka) i tačno jednog znaka '@'.
2.) neposredno levo i neposredno desno od znaka '@' mora se nalaziti slovo, a tačka ne sme biti ni prvi ni poslednji znak u adresi.
Npr. adrese 'mama@ta..ta', 'm.am.a@t..a.t..a' i 'm@t' su ispravne, dok 'ma@', '.@ma.ma', '.mama@tata' i 'ma.ma@tata.tata.' nisu.
Zaštita od spam-a je sledeća:
1.) prvo se znak '@' zameni nizom znakova 'at'
2.) zatim se niz znakova 'nospam' doda tačno jednom ili nijednom bilo gde u adresu (može ili na početak ili na kraj)
Na taj način iz originalne e-mail adrese dobijamo zaštićenu.
Napišite program koji će za neku zadanu zaštićenu e-mail adresu odrediti sve moguće različite originalne e-mail adrese od kojih se ona može dobiti.
U prvom redu ulaza se nalazi niz maksimalne dužine 100 znakova. Taj niz predstavlja zaštićenu e-mail adresu.
Treba ispisati sve različite originalne e-mail adrese (bilo kojim redosledom) iz kojih je mogla nastati zadana zaštićena adresa.
Svaku adresu ispisati u novi red.
PRIMER
ulaz
natva
izlaz
n@va

ulaz
a.batc.dnospam
izlaz
a.b@c.dnospam
a.b@c.d

REŠENJE

```
#include <algorithm>
#include <iostream>
#include <iterator>
#include <set>
#include <string>

using namespace std;
```

```
set<string> rez;

void check( string a ) {
  if( a.find( "@"  ) == string::npos ) return;
  if( a.find( ".@" ) != string::npos ) return;
  if( a.find( "@." ) != string::npos ) return;
  if( a[0] == '.' || a[a.size()-1] == '.' ) return;
  if( a[0] == '@' || a[a.size()-1] == '@' ) return;
  rez.insert( a );
}

void rec( string a, string b, int at, int nospam ) {
  if( at > 1 || nospam > 1 ) return;

  if( !b.size() ) return check( a );

  if( b.substr(0,2) == "at"       ) rec( a+"@", b.substr(2), at+1, nospam   );
  if( b.substr(0,6) == "nospam"   ) rec( a    , b.substr(6), at  , nospam+1 );
  if( b.substr(0,8) == "anospamt" ) rec( a+"@", b.substr(8), at+1, nospam+1 );
  rec( a + b.substr(0,1), b.substr(1), at, nospam );
}

int main( void ) {
  string s;
  cin >> s;

  rec( string(), s, 0, 0 );

  copy( rez.begin(), rez.end(), ostream_iterator<string> (cout,"\n") );
  return 0;
}
```

3. Dva URL-a su jednaka akko oba URL-a ukazuju na isti resurs na istom host-u, port-u, i putanji. Na primer, sledeci parovi URL-a ukazuju na isti resurs (datoteku):
http://www.ncsa.uiuc.edu/HTMLPrimer.html#GS
http://www.ncsa.uiuc.edu/HTMLPrimer.html#HD
ili
https://www.oreilly.com/index.html#p1
https://www.oreilly.com/index.html#q2
ili
http://www.ibiblio.org
http://ibiblio.org
Napisati Java program koji za 2 URL-a koji se zadaju kao parametri komandne linije ispisuje poruku DA ako URL-ovi jednaki u smislu gornje definicije. U protivnom ispisuje poruku NE.

REŠENJE

Fragment koda koji Vas je zbunjivao

```
try {
URL u1 = new URL("http://www.ncsa.uiuc.edu/HTMLPrimer.html#GS");
URL u2 = new URL("http://www.ncsa.uiuc.edu/HTMLPrimer.html#HD");
if (u1.sameFile(u2)) {
   System.out.println("DA\n");  }
else {
   System.out.println("NE\n");  }
}
catch (MalformedURLException ex) {  System.err.println(ex); }
```

4.

1. Koja IPv4 klasa adresa obezbeđuje najviše mreža?
Odgovor: Klasa C

2. Koja tvrđenja tačno opisuje klase IP adresa?
Na osnovu najstarijeg 1.bita može jasno da se odredi klasa IP adresa.
Broj bitova za identifikaciju hostova je fiksno određen klasom mreže.
Samo adrese klase A se mogu reprezentovati sa najstarijia 3 bita 100.
Najviše 24 bita određuju identitet hosta u klasi C adrese.
Najviše 24 bita određuju identitet mrežni ID.
Tri od 5 klasa adresa su rezervisane za multicast i experimentalnu upotrebu.
Odgovori:

Nije dovoljan najstariji 1.bit za jasno razlikovanje svih 5 klasa IP adresa.
Na osnovu klase znamo koliko koristi bitova za mrezu a koliko za host (32 - mreza)
U klasama A i B je potrebno manje od 24 bita, u klasi C je 24 bita potrebno za odre]ivanje hosta.

3. Kompanija ABC koristi mrežnu adresu 192.168.4.0 i masku 255.255.255.224 radi kreiranje podmreža. Koji je najveći broj hostova koji se može koristiti u svakoj podmreži?

Odgovor: 30
255.255.255.224

224 binarno = 11100000
znaci ostaje nam 5 bitova za host i onda imamo 2^5 razlicitih hostova gde spadaju i 2 specijalna za broadcast i rip pa je resenje 32-2 = 30

4. Koja je mrežna broadcast adresa u klasi C za adresu 192.168.32.0 sa podraumevanom mrežnom podmaskom?

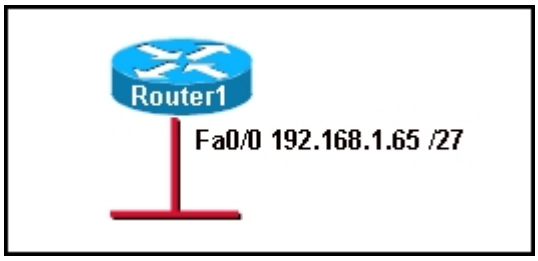
Odgovor: 192.168.32.255

5. Za koiko bitova IPv6 adrese povećavaju veličinu IPv4 adrese?
Odgovor: 128-32=96

6. Koji je opseg prvog okteta u klasi B adresa?
Odgovor: Od 128 do 191

7. Koja IPv4 klasa adresa obezbeđuje najviše host adresa po mreži?
Odgovor: Klasa A

8. Koji segment IP adresa dozvoljava hostovima koji su povezani na Router1 Fa0/0 da pristupe mrežama spolja?

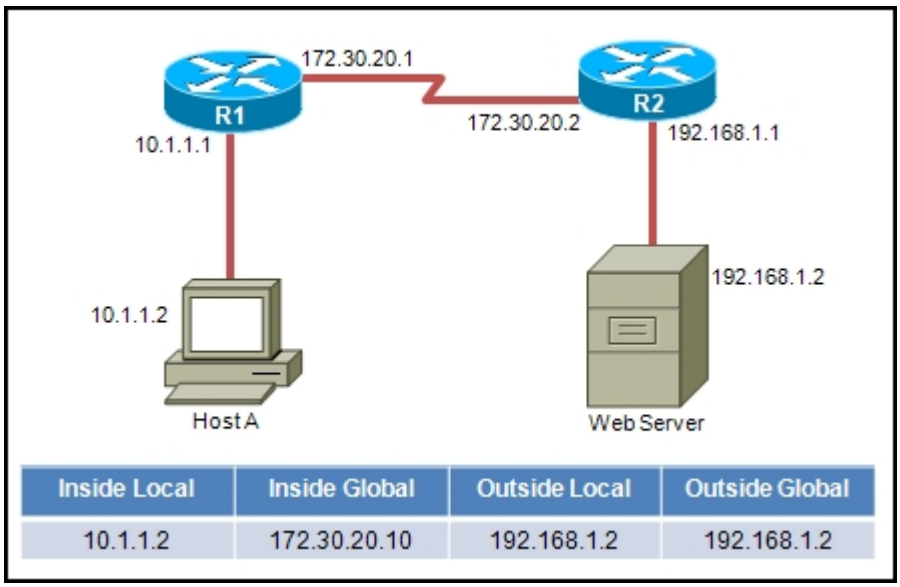


Odgovor: 192.168.1.66 do 192.168.1.94

192.168.1.65 = **11000000. 10101000.00000001. 010**00001

192.168.1.95=**11000000. 10101000.00000001. 010**11111

9. Ruter R1 unutar mreže radi NATovanje adresa za rang adresa 10.1.1.0/24. HostA je poslao zahtev web serveru. Koja je odredišna IP adresa paketa koji se vraća sa web servera?



Inside Local	Inside Global	Outside Local	Outside Global
10.1.1.2	172.30.20.10	192.168.1.2	192.168.1.2

170.30.20.10