Rešenja:

1. OSTATAK

Potrebni moduli za dostizanje rešenja zadatka:

• pronaći proste brojeve ne veće od *n*;

Koristimo Eratostenovo sito i funkcija Eratosten(n) vraća 1 ako n je prost broj, a 0 u ostalim slučajevima

• pronaći najmanji prost broj *p* veći od *n*;

Proverimo za svaki prirodam broj veći od n da li je prost broj sve dok ne nađemo broj *p*.

• pronaći ostatak pri deljenju broja *S*(*n*) brojem *p*.

Koristimo niz *s* tako da za *i* = 1, 2, …, *n* važi da *s*[*i*] je ostatak pri deljenju broja načina na koji se broj *i* može zapisati pomoću zbira prostih brojeva sa brojem *p*; *s*[*i*] se može dobiti iterativno u *k* koraka (*k* = 0, 1, …, *brojac* – 1).

```
#include <iostream>
using namespace std;
#define MAXN 50010

bool sito[MAXN];
long a[MAXN];
long s[MAXN];
long brojac;

int Eratosten(long n)
{ for(long i = 2; i <= n; i++)
    if (!sito[i])
      { a[brojac] = i;
        brojac++;
        for(long j = i+i; j <= n; j+= i)
          sito[j] = true;
      }
  if (a[brojac-1] == n) return 1;
  return 0;
}

int main()
{  long n;
   cin >> n;
```

```
    if (n == 1)
      { cout << 0 << endl;
        return 0;
      }

    int n_je_prost;
    n_je_prost = Eratosten(n);
    long p = n+1;
    bool p_je_prost;
    do
      { p_je_prost= true;
        long i = 0;
        while(p_je_prost&& a[i]*a[i] <= p)
          { if (p%a[i] == 0)
              {p_je_prost= false;
               p++;
              }
            i++;
          }
        }
    while(!p_je_prost);

    s[0] = 1;
    for(long k = 0; k < brojac; k++)
      { long x = a[k];
        for(long i = x; i <= n; i++)
          s[i] = (s[i]+s[i-x])%p;
      }

    s[n] = (s[n]+p-n_je_prost)%p;
    cout << s[n] << endl;

    return 0;
  }
```

2. PRAVOUGAONIK

Neka su dužine stranica pravougaonika P redom a i b i neka $d = \text{NZD}(a,b)$. Onda važi da $f(P) = a + b - d.$

Dakle, potrebno je naći broj celobrojnih rešenja jednačine $a + b - d = N$, tako da $a \leq b$.

Neka $a = x*d$, $b = y*d$. Tada je $1 \le x \le y$ i NZD(x, y) = 1.
Napisimo jednacinu oblika $(x + y - 1)\, d = N$.
Dakle, d deli N, te $N = m*d$. Dalje vazi da $x + y = m + 1$.

Dakle za svako m , koje deli N *potrebno* je pronaci broj celobrojnih resenja jednacine $x + y = m + 1$, tako da $1 \le x \le y$ i NZD(x, y) = 1.
Vazi da NZD(x, y) = NZD(x,m+1− x) = NZD(x,m+1).

```
#include<iostream>

using namespace std;
int nzd(int a, int b)
{ int r = a%b;
while(r > 0)
{ a = b; b = r; r = a%b; }
return b;
}

int main()
{ int n;
cin >> n;
int s = 0;
for(int m = 1; m*m <= n; m++)
{ if (n % m == 0)
    if (m*m == n)
{ for(int x = 1; x <= (m + 1)/2; x++)
if (nzd(x, m + 1) == 1) s++;
}
else
{ for(int x = 1; x <= (m + 1)/2; x++)
if (nzd(x, m + 1) == 1) s++;
for(int x = 1; x <= (n/m + 1)/2; x++)
if (nzd(x, n/m + 1) == 1) s++;
}
}
cout << s << endl;
return 0;
}
```

3. INVERZIJA

Rešenje:

Analiza rešenja:

Naivno rešenje je direktna provera za svaki par indeksa i, j (tako da $i<j$) da li važi a[i] > a[j].

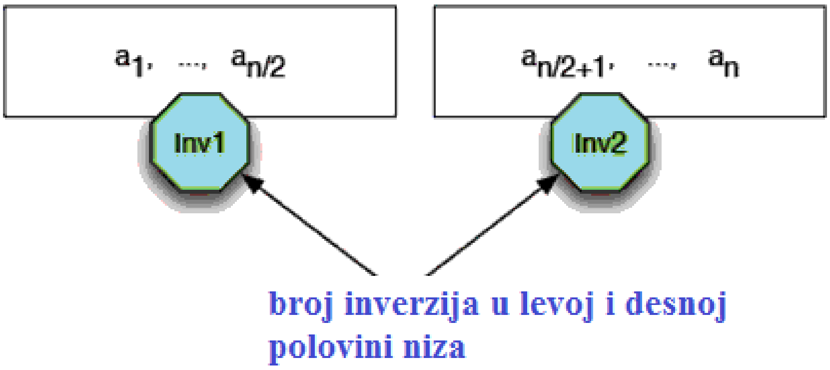
```
int InvCount(int a[], int n)

{

  int inv_count = 0;

  int i, j;


  for(i = 0; i < n - 1; i++)

    for(j = i+1; j < n; j++)

      if(a[i] > a[j])   inv_count++;


  return inv_count;

}
```

Ova direktna provera ima vremensku složenost $O(n^2)$ što je previše sporo, s obzirom na vremensko ograničenje u ovom zadatku.

Naime, kako je $n \leq 1000000$, potrebno je voditi računa da vremenska složenost rešenja $O(n^2)$ ne bi bila odgovarajuća za veliko n, tj. velike nizove.

Prebrojane inverzije pokazuju u kojoj meri je niz predsortiran. Ako je niz sortiran (u konkretnom zadatku u rastućem poretku), onda broj inverzija je 0. Ako je niz monotono opadajući, onda je broj inverzija maksimalan.

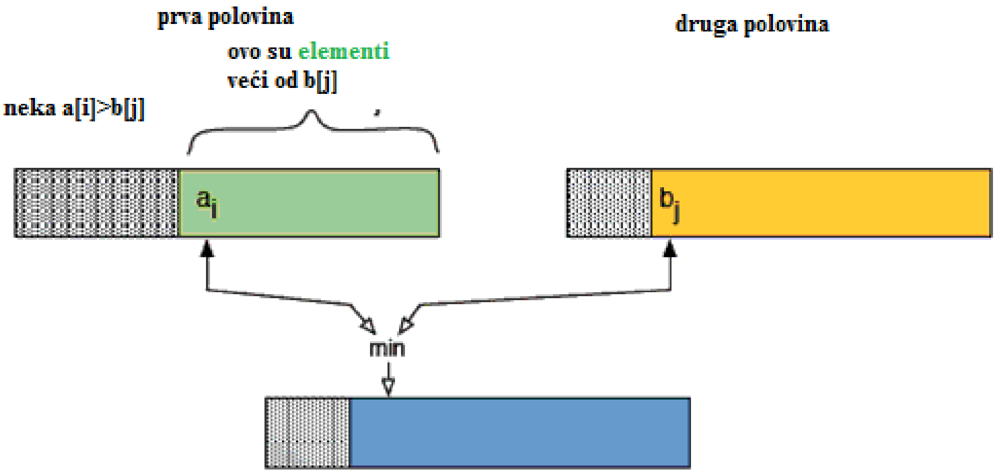
Pretpostavimo da znamo broj inverzija u levoj i desnoj polovini niza (sačuvanih redom u promenljivama inv1 i inv2). Koje inverzije eventualno nisu uračunate u zbiru inv1 + inv2?



Nisu uračunate verzije koje se pojavljuju pri spajanju polovina nizova.

Dakle, ukupan broj inverzija je jednak zbiru broju inverzija u levom i desnom podnizu i inverzija pri spajanju podnizova.

U procesu spajanja, neka se indeks i koristi za indeksiranje levog podniza i neka se j koristi za indeksiranje desnog podniza. U bilo kom koraku u modulu spajanja spojUredi(), ako važi da a[i] je veće od a[j] za i<j, onda postoji (sredina – i) inverzija, jer su levi i desni podniz sortirani, te preostali elementi u levom podnizu (a[i+1], a[i+2] … a[sredina]) su veći od a[j]




```
#include <iostream>
```

```
#include <stdio.h>
using namespace std;
int n, a[1000000];
long long r;
void spojUredi(int l, int d)
{    if(l>=d)      return;
     int s=(l+d)/2;
    static int b[1000000];
    int i=l,j=s+1,k=0;
    spojUredi(l,s);
    spojUredi(s+1,d);
   while(i<=s&&j<=d)
        if(a[i]<a[j])           b[k++]=a[i++];
        else
       {       b[k++]=a[j++];
              r+=s-i+1;
       }
    while (i<=s)        b[k++]=a[i++];
    while (j<=d)        b[k++]=a[j++];
    for(int p=0;p<=d-l;p++)    a[p+l]=b[p];
}
int main()
{    scanf("%d", &n);
     for(int i=0;i<n;i++)    scanf("%d", &a[i]);
    spojUredi(0,n-1);
    printf("%lld", r);
}
```