

15. tekmovanje ACM v znanju računalništva za srednješolce

28. marca 2020

NALOGE ZA DRUGO SKUPINO

Naloge rešuj samostojno; ne sprašuj drugih ljudi za nasvete ali pomoč pri reševanju (niti v živo niti prek interneta ali kako drugače), ne kopiraj v svoje odgovore tuje izvorne kode in podobno. Tekmovalna komisija si pridržuje pravico, da tekmovalce diskvalificira, če bi se kasneje izkazalo, da nalog niso reševali sami. Internet lahko uporabljaš, če ni v nasprotju s prejšnjimi omejitvami (npr. za branje dokumentacije), vendar za reševanje nalog ni nujno potreben. Tvoje odgovore bomo pregledali in ocenili ročno, zato manjše napake v sintaksi ali pri klicih funkcij standardne knjižnice niso tako pomembne, kot bi bile na tekmovanjih z avtomatskim ocenjevanjem.

Tekmovanje bo potekalo na strežniku <https://rtk.fri.uni-lj.si/>, kjer dobiš naloge in oddajaš svoje odgovore. Uporabniška imena in gesla (bo)ste dobili po elektronski pošti. Pri oddaji preko računalnika rešitev natipkaš neposredno v brskalniku. Med tipkanjem se rešitev na približno dve minuti samodejno shrani. Poleg tega lahko sam med pisanjem rešitve izrecno zahtevaš shranjevanje rešitve s pritiskom na gumb „Shrani spremembe“. Gumb „Shrani in zapri“ uporabiš, ko si bodisi zadovoljen z rešitvijo ter si zaključil nalogo, ali ko želiš začasno prekiniti pisanje rešitve naloge ter se lotiti druge naloge. Po pritisku na ta gumb se vpisana rešitev shrani in te vrne v glavni menu. (Oddano rešitev lahko kasneje še spreminjaš.) Za vsak slučaj priporočamo, da pred oddajo shraniš svoj odgovor tudi v datoteko na svojem lokalnem računalniku.

Med reševanjem lahko vprašanja za tekmovalno komisijo postavljaš prek zasebnih sporočil na tekmovalnem strežniku (ikona oblaka zgoraj desno), izjemoma pa tudi po elektronski pošti na rtk-info@ijs.si.

Če imaš pri oddaji odgovorov prek spletnega strežnika kakšne težave, lahko izjemoma pošlješ svoje odgovore po elektronski pošti na rtk-info@ijs.si, vendar nas morajo doseči pred koncem tekmovanja; odgovorov, prejetih po koncu tekmovanja, ne bomo upoštevali.

Svoje odgovore dobro utemelji. Če pišeš izvorno kodo programa ali podprograma, **OBVEZNO** tudi v nekaj stavkih z besedami opiši idejo, na kateri temelji tvoja rešitev. Če ni v nalogi drugače napisano, lahko tvoje rešitve predpostavljajo, da so vhodni podatki brez napak (da ustrezajo formatu in omejitvam, kot jih podaja naloga). Zaželeno je, da so tvoje rešitve poleg tega, da so pravilne, tudi učinkovite; bolj učinkovite rešitve dobijo več točk (s tem je mišljeno predvsem, naj ima rešitev učinkovit algoritem; drobne tehnične optimizacije niso tako pomembne). **Nalog je pet** in pri vsaki nalogi lahko dobiš od 0 do 20 točk.

Rešitve bodo objavljene na <http://rtk.ijs.si/>. Predvidoma nekaj dni po tekmovanju bodo tam objavljeni tudi rezultati.

15. tekmovanje ACM v znanju računalništva za srednješolce

28. marca 2020

NASVETI ZA 1. IN 2. SKUPINO

Nekatere naloge so tipa **napiši program** (ali **napiši podprogram**), nekatere pa tipa **opiši postopek**. Pri slednjih ti ni treba pisati programa ali podprograma v kakšnem konkretnem programskem jeziku, ampak lahko postopek opišeš tudi kako drugače: z besedami (v naravnem jeziku), psevdokodo (glej spodaj), diagramom poteka itd. Glavno je, da je tvoj opis dovolj natančen, jasen in razumljiv, tako da je iz njega razvidno, da si dejansko našel in razumel pot do rešitve naloge.

Psevdokodi pravijo včasih tudi strukturirani naravni jezik. Postopek opišemo v naravnem jeziku, vendar opis strukturiramo na podoben način kot pri programskih jezikih, tako da se jasno vidi strukturo vejitev, zank in drugih programskih elementov.

Primer opisa postopka v psevdokodi: recimo, da imamo zaporedje besed in bi ga radi razbili na več vrstic tako, da ne bo nobena vrstica preširoka.

```
naj bo trenutna vrstica prazen niz;
pregleduj besede po vrsti od prve do zadnje:
    če bi trenutna vrstica z dodano trenutno besedo (in presledkom
    pred njo) postala predolga,
        izpiši trenutno vrstico in jo potem postavi na prazen niz;
        dodaj trenutno besedo na konec trenutne vrstice;
    če trenutna vrstica ni prazen niz, jo izpiši;
```

(Opomba: samo zato, ker je tu primer psevdokode, to še ne pomeni, da moraš tudi ti pisati svoje odgovore v psevdokodi.)

Če pa v okviru neke rešitve pišeš izvorno kodo programa ali podprograma, obvezno poleg te izvorne kode v nekaj stavkih opiši, kako deluje (oz. naj bi delovala) tvoja rešitev in na kakšni ideji temelji.

Pri ocenjevanju so vse naloge vredne enako število točk. Svoje odgovore dobro utemelji. Prizadevaj si predvsem, da bi bile tvoje rešitve pravilne, ob tem pa je zaželeno, da so tudi čim bolj učinkovite; take dobijo več točk kot manj učinkovite (s tem je mišljeno predvsem, naj ima rešitev učinkovit algoritem; drobne tehnične optimizacije niso tako pomembne). Za manjše sintaktične napake se ne odbije veliko točk. Priporočljivo in zaželeno je, da so tvoje rešitve napisane pregledno in čitljivo.

Če naloga zahteva branje ali obdelavo vhodnih podatkov, lahko tvoja rešitev (če v nalogi ni drugače napisano) predpostavi, da v vhodnih podatkih ni napak (torej da je njihova vsebina in oblika skladna s tem, kar piše v nalogi).

Nekatere naloge zahtevajo branje podatkov s standardnega vhoda in pisanje na standardni izhod. Za pomoč je tu nekaj primerov programov, ki delajo s standardnim vhodom in izhodom:

- Program, ki prebere s standardnega vhoda dve števili in izpiše na standardni izhod njuno vsoto:

```
program BranjeStevil;
var i, j: integer;
begin
  ReadLn(i, j);
  WriteLn(i, ' + ', j, ' = ', i + j);
end. {BranjeStevil}

#include <stdio.h>
int main() {
  int i, j; scanf("%d %d", &i, &j);
  printf("%d + %d = %d\n", i, j, i + j);
  return 0;
}
```

- Program, ki bere s standardnega vhoda po vrsticah, jih šteje in prepisuje na standardni izhod, na koncu pa izpiše še skupno dolžino:

```

program BranjeVrstic;
var s: string; i, d: integer;
begin
  i := 0; d := 0;
  while not Eof do begin
    ReadLn(s);
    i := i + 1; d := d + Length(s);
    WriteLn(i, ', vrstica: ', s, ', ');
  end; {while}
  WriteLn(i, ' vrstic, ', d, ' znakov. ');
end. {BranjeVrstic}

#include <stdio.h>
#include <string.h>
int main() {
  char s[201]; int i = 0, d = 0;
  while (gets(s)) {
    i++; d += strlen(s);
    printf("%d. vrstica: \"%s\\n\"", i, s);
  }
  printf("%d vrstic, %d znakov.\\n", i, d);
  return 0;
}

```

Opomba: C-jevska različica gornjega programa predpostavlja, da ni nobena vrstica vhodnega besedila daljša od dvesto znakov. Funkciji `gets` se je v praksi bolje izogibati, ker pri njej nimamo zaščite pred primeri, ko je vrstica daljša od naše tabele `s`. Namesto `gets` bi bilo bolje uporabiti `fgets`; vendar pa za rešitev naših tekmovalnih nalog v prvi in drugi skupini zadošča tudi `gets`.

- Program, ki bere s standardnega vhoda po znakih, jih prepisuje na standardni izhod, na koncu pa izpiše še število prebranih znakov (ne všteti znakov za konec vrstice):

```

program BranjeZnakov;
var i: integer; c: char;
begin
  i := 0;
  while not Eof do begin
    while not Eoln do
      begin Read(c); Write(c); i := i + 1 end;
    if not Eof then begin ReadLn; WriteLn end;
  end; {while}
  WriteLn('Skupaj ', i, ' znakov. ');
end. {BranjeZnakov}

#include <stdio.h>
int main() {
  int i = 0, c;
  while ((c = getchar()) != EOF) {
    putchar(c); if (i != '\n') i++;
  }
  printf("Skupaj %d znakov.\\n", i);
  return 0;
}

```

Še isti trije primeri v pythonu:

Branje dveh števil in izpis vsote:

```

import sys
a, b = sys.stdin.readline().split()
a = int(a); b = int(b)
print("%d + %d = %d" % (a, b, a + b))

```

Branje standardnega vhoda po vrsticah:

```

import sys
i = d = 0
for s in sys.stdin:
  s = s.rstrip('\n') # odrežemo znak za konec vrstice
  i += 1; d += len(s)
  print("%d. vrstica: \"%s\"" % (i, s))
print("%d vrstic, %d znakov." % (i, d))

```

Branje standardnega vhoda znak po znak:

```

import sys
i = 0
while True:
  c = sys.stdin.read(1)
  if c == "": break # EOF
  sys.stdout.write(c)
  if c != '\n': i += 1
print("Skupaj %d znakov." % i)

```

Še isti trije primeri v javi:

```
// Branje dveh števil in izpis vsote:
import java.io.*;
import java.util.Scanner;

public class Primer1
{
    public static void main(String[] args) throws IOException
    {
        Scanner fi = new Scanner(System.in);
        int i = fi.nextInt(); int j = fi.nextInt();
        System.out.println(i + " + " + j + " = " + (i + j));
    }
}

// Branje standardnega vhoda po vrsticah:
import java.io.*;

public class Primer2
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader fi = new BufferedReader(new InputStreamReader(System.in));
        int i = 0, d = 0; String s;
        while ((s = fi.readLine()) != null) {
            i++; d += s.length();
            System.out.println(i + ". vrstica: \"" + s + "\""); }
        System.out.println(i + " vrstic, " + d + " znakov.");
    }
}

// Branje standardnega vhoda znak po znak:
import java.io.*;

public class Primer3
{
    public static void main(String[] args) throws IOException
    {
        InputStreamReader fi = new InputStreamReader(System.in);
        int i = 0, c;
        while ((c = fi.read()) >= 0) {
            System.out.print((char) c); if (c != '\n' && c != '\r') i++; }
        System.out.println("Skupaj " + i + " znakov.");
    }
}
```