Задаци за вежбање (основни ниво)

RisingStar 01

1. Ако се зна који је дан у недељи био први дан текућег месеца, напиши програм који одређује који дан је данас.

Улаз

Прва линија стандардног улаза садржи ознаку дана у недељи првог дана текућег месеца (1 - понедељак, 2 - уторак, 3 - среда, 4 - четвртак, 5 - петак, 6 - субота, 7 - недеља), а друга линија садржи редни број текућег дана у текућем месецу.

Излаз

На стандардни излаз исписати ознаку данашњег дана.

Пример 1

Улаз

```
1
4
```

Излаз

```
4
```

Објашњење

Први дан је био понедељак, па је четврти дан четвртак.

Пример 2

Улаз

```
1
8
```

Излаз

```
1
```

Објашњење

Први дан је био понедељак, па је и осми дан понедељак.

Пример 3

Улаз

```
3
17
```

Излаз

```
5
```

Објашњење

Први дан је била среда, па је седамнаести дан петак.

Решење:
На ознаку првог дана у месецу додаћемо колико је дана прошло до данашњег.
Тај број протеклих дана се добија тако што се од редног броја данашњег дана одузме 1.
Пошто у обзир треба узети и периодично понављање дана, морамо пронаћи остатак при дељењу са 7.
Како се дани броје од 1 до 7, а остаци при дељењу са 7су од 0 до 6, примењујемо “алгебарски трик” да пре израчунавања остатка одузмемо 1, а након израчунавања додамо 1.

```
#include <iostream>

using namespace std;

int main() {
  int prvi;
  cin >> prvi;
  int danas;
  cin >> danas;
  cout << (prvi + (danas - 1) - 1) % 7 + 1 << endl;
  return 0;
}
```

---

2. Кућа има по један прозор на источној, јужној и западној страни. У кући је ваздух устајао ако није отворен ни један прозор, кућа се проветрава ако је отворен један или два прозора, а промаја је ако су отворена сва три прозора. Одредити шта се тренутно дешава у кући на основу тога који прозори су отворени.

Улаз

Улаз се састоји од 3 реда, а у сваком реду је реч `DA` или `NE`, према томе да ли је отворен прозор на једној страни куће. Дате речи се редом односе на прозоре на источној, јужној и западној страни.

Излаз

Један од текстова `promaja`, `vetrenje`, или `ustajao vazduh` који описује стање у кући.

Пример 1

Улаз

```
DA
NE
DA
```

Излаз

```
vetrenje
```

Пример 2

Улаз

```
NE
NE
NE
```

Излаз

```
ustajao vazduh
```

Пример 3

Улаз

```
DA
DA
DA
```

Излаз

```
promaja
```

Решење:
Задатак се може решити гранањем, тј. наредбом селекције if-**else**.
Проверу да ли су сва три прозора отворена можемо извршити оператором “логичко и”, проверу да ли је бар један прозор отворен можемо извршити оператором “логичко или”.

```
#include <iostream>

using namespace std;

int main() {
```

```
    string s;
    cin >> s; bool istok = (s == "DA");
    cin >> s; bool jug = (s == "DA");
    cin >> s; bool zapad = (s == "DA");

    if (istok && jug && zapad)
        cout << "promaja" << endl;
    else if (istok || jug || zapad)
        cout << "vetrenje" << endl;
    else
        cout << "ustajao vazduh" << endl;
    return 0;
}
```

---

3. У соби постоје плава и жута сијалица. За сваку сијалицу је познато време (у облику сат, минут, секунд) када је укључена и када је искључена. Напиши програм који одређује колико времена је соба била осветљена плаво, колико времена је била осветљена зелено и колико времена је била осветљена жуто (соба је осветљена зелено док су истовремено укључене плава и жута сијалица).

Улаз

Са стандардног улаза се уносе четири реда са по три броја раздвојена размацима (сат, минут и секунд укључивања плаве сијалица, сат, минут и секунд њеног искључивања, сат, минут и секунд укључивања жуте сијалице и сат, минут и секунд њеног искључивања).

Излаз

У првој линији стандардног излаза исписати три броја раздвојена двотачкама која представљају време (у сатима, минутима и секундама) које је соба била обојена плаво, у наредној линији у истом облику исписати време које је соба била обојена зелено и у последњој линији у истом облику исписати време које је соба била обојена жуто.

Пример 1

Улаз

```
13 15 0
14 20 0
13 45 0
15 30 0
```

Излаз

```
0:30:0
0:35:0
1:10:0
```

Пример 2

Улаз

```
10 50 20
12 40 11
8 45 30
10 30 20
```

Излаз

```
1:49:51
0:0:0
1:44:50
```

Решење:
Згодно је да током програма улазне податаке са временске осе прерачунамо у најмању јединицу (у овом случају у секунде) и да током рада користимо целобројне вредности временског тренутка.

Најједноставније је одредити дужину трајања зеленог светла. Применићемо технику одређивања дужине пресека два интервала. Зелено светло траје од тренутка када је друго светло укључено па до тренутка када је прво светло искључено (под претпоставком да је друго светло укључено пре него што је прво искључено). Пошто распоред укључивања и искључивања плавог и жутог светла може бити произвољан, време укључивања другог светла можемо израчунати као веће од времена укључивања плавог и жутог светла, док се време искључивања првог светла може израчунати као мање од времена искључивања плавог и времена искључивања жутог светла. Трајање зеленог светла се може израчунати као разлика ова два времена, при чему је то трајање 0, ако се прво светло искључило пре него што се друго укључило (ово се једноставно може израчунати тако што се узме већа од вредности 0 и разлике између времена искључивања првог и укључивања другог светла).

Када се зна трајање зеленог светла, трајање плавог тј. жутог светла се може веома једноставно израчунати тако што се од времена сијања плаве тј. жуте сијалице одузме трајање зеленог светла. Време сијања плаве тј. жуте сијалице се може израчунати веома лако, тако што се од времена њеног искључивања одузме време њеног укључивања.

Решење (које користи потпрограм)
Поразмислити о имплементацији без потпрограма:

```
#include <iostream>
#include <algorithm>

using namespace std;

// prevodi vreme dato u broju sati, minuti i sekundi u vreme dato u
// obliku broja sekundi proteklih od pocetka dana (tj. prethodne
// ponoci)
int uSekunde(int h, int m, int s) {
  return h*60*60 + m*60 + s;
}

// prevodi vreme dato u broju sekundi od pocetka dana u vreme izrazeno
// u obliku sati, minuti i sekundi
void odSekundi(int S, int& h, int& m, int& s) {
```

```
  s = S % 60;
  m = (S / 60) % 60;
  h = (S / (60*60));
}

// ucitava vreme dato u obliku sati, minuti i sekundi
// i vraca broj sekundi
int ucitajVreme() {
  int h, m, s;
  cin >> h >> m >> s;
  return uSekunde(h, m, s);
}

// stampa vreme dato u broju sekundi od pocetka dana u obliku sati,
// minuti i sekundi
void stampajVreme(int S) {
  int h, m, s;
  odSekundi(S, h, m, s);
  cout << h << ":" << m << ":" << s << endl;
}

int main() {
  // Vreme ukljucivanja i iskljucivanja plave sijalice u sekundama
  int ukljucPlava_S  = ucitajVreme();
  int iskljucPlava_S = ucitajVreme();
  // Vreme ukljucivanja i iskljucivanja zute sijalice u sekundama
  int ukljucZuta_S   = ucitajVreme();
  int iskljucZuta_S  = ucitajVreme();

  // Vreme u sekundama ukljucivanja druge sijalice
  int ukljucDruga = max(ukljucPlava_S, ukljucZuta_S);
  // Vreme u sekundama iskljucivanja prve sijalice
  int iskljucPrva = min(iskljucPlava_S, iskljucZuta_S);
  // Trajanje zelenog svetla u sekundama
  int zelenoSvetlo = max(iskljucPrva - ukljucDruga, 0);
  // Trajanje plave sijalice u sekundama
  int plavaSijalica = iskljucPlava_S - ukljucPlava_S;
  // Trajanje plavog svetla u sekundama
  int plavoSvetlo  = plavaSijalica - zelenoSvetlo;
  // Trajanje zute sijalice u sekundama
  int zutaSijalica = iskljucZuta_S - ukljucZuta_S;
  // Trajanje zutog svetla u sekundama
  int zutoSvetlo   = zutaSijalica - zelenoSvetlo;

  // Prevodimo trajanje svakog od tri svetla u sate, minute i sekunde
  // i ispisujemo rezultat
  int h, m, s;
  stampajVreme(plavoSvetlo);
  stampajVreme(zelenoSvetlo);
  stampajVreme(zutoSvetlo);
  return 0;
}
```

==========

4. Напиши програм који одређује колико је потеза краљу на шаховској табли потребно да дође на дато поље. Краљ се у сваком потезу може померити за једно поље у било ком од 8 смерова (горе, доле, лево, десно и дијагонално).

Улаз

Са стандардног улаза се учитавају четири броја између 1 и 8 (у свакој линији по два, раздвојена размаком). Прва два броја представљају координате полазног, а друга два броја координате долазног поља.

Излаз

На стандардни излаз исписати један цео број који представља најмањи број потеза потребан краљу да са полазног стигне до долазног поља.

Пример

Улаз

```
1 2
3 4
```

Излаз

```
2
```

Пример

Улаз

```
8 3
2 5
```

Излаз

```
6
```

Решење:
Нека су (x1,y1) координате полазног поља, а (x2,y2) координате долазног поља. Растојање по првој координати једнако је |x2−x1|, а по другој координати једнако је |y1−y2|.
Да би краљ стигао на жељено поље, оба ова растојања морају бити смањена на нулу. У сваком кораку (било хоризонталном, било вертикалном, било дијагоналном) свака од координата се може променити највише за један (у дијагоналном потезу истовремено се мењају обе координате, али свака за по један). Зато се у сваком од потеза хоризонтално и вертикално растојање могу смањити за један. У почетку краљ може ићи дијагонално истовремено смањујући оба растојања за по један, све док једно од растојања не постане једнако нули (док не стигне у врсту или колону у којој се налази циљно поље). Након тога може се кретати хоризонтално тј. вертикално до циља.
Дакле, број потеза потребан да се стигне до циљаног поља је једнак већем од растојања по координатама x и y тј. једнако је вредности max(|x2−x1|,|y2−y1|).
Максимум два броја се може израчунати било гранањем, било библиотечком функцијом.
Апсолутну вредност је могуће израчунати коришћењем функције

abs
која је за целе бројеве декларисана у заглављу

<cstdlib>

```
#include <iostream>
#include <cstdlib>
#include <algorithm>

using namespace std;

int main() {
  // ucitavamo koordinate dva polja
  int x1, y1, x2, y2;
  cin >> x1 >> y1 >> x2 >> y2;
  // odredjujemo i ispisujemo najmanji broj poteza kralja
  cout << max(abs(x1 - x2), abs(y1 - y2)) << endl;
  return 0;
}
```

==========

5. Пера и Мика живе у истој улици. Микина кућа је удаљенија од школе. Они иду у школу истим путем, полазе из куће у исто време и равномерно се крећу. Пера се креће брзином v1 m/s , а Мика брзином v2 m/s (v2>v1). Написати програм којим се одређује колико је растојање између њихових кућа, ако се зна да је после t секунди Мика био d метара иза Пере.

Улаз

Са стандардног улаза се уносе четири реална броја, сваки у посебној линији. Они редом представљају брзину кретања Пере (v1) и брзину кретања Мике (v2), изражене у ms, потом број секунди (t) и растојање између Пере и Мике у метрима (d).

Излаз

На стандарном излазу приказати реалан број, на две децимале, који представља колико је растојање између њихових кућа.

Пример

Улаз

```
1.6
2.1
10
30.0
```

Излаз

```
35.00
```

Решење: У задатку се обојица крећу равномерно, тако да ће се решење заснивати на вези између брзине, пређеног пута и времена

```
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
  double v1, v2, d, t;
  cin >> v1 >> v2 >> t >> d;
  double s1 = v1 * t;       // put koji predje Pera
  double s2 = v2 * t;       // put koji predje Mika
  double x  = s2 - s1 + d;  // rastojanje izmedju njihovih kuca
  cout << fixed << showpoint << setprecision(2) << x << endl;
  return 0;
}
```