Увод у програмирање, I смер - Јануар 1 - 2025

1. У фабрици рубикових коцки радници морају да запакују коцке у кутије једнаких димензија. Свака коцка има ивицу дужине 1 дециметар. Кутије су облика квадра, њихове димензије су изражене у дециметрима и природни су бројеви.

   Са стандардног улаза се најпре учитавају дужина, ширина и висина једне кутије, а затим и број рубикових коцки. На стандардни излаз исписати број кутија потребних да се спакују све коцке, под условом да радници хоће да искористе минималан број кутија. Претпоставити да је улаз исправно задат.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
Улаз:	Улаз:	Улаз:	Улаз:
2	2	3	2
2	3	1	3
2	4	5	2
17	44	54	100
Излаз:	Излаз:	Излаз:	Излаз:
3	2	4	9

   Решење: Након што учитамо димензије кутије и број коцки рачунамо запремину кутије (формулом $V = a \cdot b \cdot c$). Број кутија потребних да би стале све коцке јединичне димензије рачунамо тако што број коцки поделимо са запремином кутије (у кутију запремине V може да стане V коцки) и додајемо 1 уколико је дељење са остатком јер то значи да не стаје у тачан број кутија који смо добили целобројним дељењем. Алтернативно решење је претварање вредности у реалне и употреба `ceil()` функције која заокружи вредност на прву већу целу.

```
#include <iostream>

using namespace std;

int main() {
    int a, b, c, n;
    cin >> a >> b >> c >> n;

    int zapKutije = a * b * c;
    int rez = n / zapKutije + (n % zapKutije != 0 ? 1 : 0);

    cout << rez << endl;

    return 0;
}
```

2. Једном играчу у игри јамб остала су два непопуњена поља. У питању су поље за ручни јамб и ручни фул. Како би играч могао да упише поена у било која од та два поља, мора да баци свих 5 коцкица одједном. Како би уписао поене у поље за

јамб мора добити исти број на свих 5 коцкица. Тада на збир свих коцкица додаје 50 поена и уписује тај број поена у поље. Како би уписао вредност на фул, потребно је да добије исту вредност на 3 коцкице и неку другу вредност на преостале две. Тада се на збир добијених вредности додаје 30 и уписује се у поље.

Са стандардног улаза учитава се пет бројева добијених играчевим бацањем, уређених неопадајуће. На стандардни излаз исписати број поена које играч уписује након бацања или 0 уколико нема шта да упише у своја поља. У случају неисправног улаза исписати -1 на излаз.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
Улаз:	Улаз:	Улаз:	Улаз:
2	1	1	-2
2	1	3	2
2	1	3	2
5	1	3	4
5	1	6	4
Излаз:	Излаз:	Излаз:	Излаз:
46	55	0	-1

Решење: Након што учитамо свих пет вредности проверимо да ли су вредности између један и шест (једине вредности које могу да буду на коцки јер коцка има шест страница), уколико није променљиву у којој чувамо резултат променимо на -1. Уколико су све вредности једнаке у питању је поље за ручни јамб где се уписује вредност и тада нам је вредност збир свих вредности сабран са 50. Последњи случај нам је када је у питању ручни фул. У поставци задатка речено је како су вредности дате неопадајуће што значи да две различите вредности могу бити само прве две или последње две када су остале три једнаке једна другој. Тако проверимо да ли је ручни фул у оба случаја ако јесте резултат нам је збир броја 30 и свих вредности. Могуће је елегантније решити задатак употребом низа и петљи, али с обзиром на то да има само пет вредности нема потребе.

```
#include <iostream>

using namespace std;

int main() {
    int a, b, c, d, e;
    cin >> a >> b >> c >> d >> e;

    int rez = 0;
    if(a <= 0 || a > 6 || b <= 0 || b > 6 || c <= 0 || c > 6
    || d <= 0 || d > 6 ||  e <= 0 || e > 6) rez = -1;
    else if(a == b && b == c && c == d && d == e) rez = 50 + a + b + c + d + e;
    else if((a == b && b == c && d == e) ||
    (a == b && c == d && d == e)) rez = 30 + a + b + c + d + e;

    cout << rez << endl;
```

```
    return 0;
}
```

3. Деца су спремила посебна правила за игру школице. Школица има 10 поља, нумерисаних бројевима од 1 до 10. Играч креће са поља број 1 и циљ му је да стигне на поље број 10. Креће се бацањем коцкице све док не добије број 1. Након сваког бацања, помера се напред за онолико поља колико је добио на коцкици. Ако стане на парно поље остаје ту где јесте. Ако стане на непарно поље, помера се једно поље назад. Ако би добијеним бројем прескочио поље са бројем 10, остаје на пољу на којем је био пре бацања. Када играч добије број 1 игра се завршава и проверава се да ли се налази на пољу број 10. Уколико се налази, победио је, иначе изгубио је.

   Са стандардног улаза учитавају се бројеви добијени на коцкици све док се не добије број 1. На стандардни излаз исписати редни број бацања у коме је играч стао на поље број 10 или 0. Бацања се броје од 1, претпоставити да је улаз исправно задат.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
Улаз: 4 3 4 2 1 Излаз: 3	Улаз: 6 5 1 Излаз: 0	Улаз: 2 3 6 3 3 1 Излаз: 3	Улаз: 5 3 4 5 2 6 2 1 Излаз: 5

   Решење: Играч креће од позиције број 1, чувамо такође број корака и вредност бацања. У `while` петљи учитавамо све са стандардног улаза. На почетку увећавамо број корака, и излазимо из петље уколико је бачена јединица. Ако је бачена вредност таква да би позиција била већа од 10 прескачемо остатак петље, а уколико смо стигли до поља 10 прекидамо рад петље јер остала бацања нису важна, такође дајемо тачну вредност променљивој крај да бисмо знали да је играч дошао до позиције број 10. У остатку тела петље додајемо вредност бацања на позицију и смањујемо је уколико је позиција непарног броја.

```
#include <iostream>

using namespace std;

int main() {
    int pozicija = 1;
    int korak = 0;
    bool kraj = false;
    int bacanje;

    while(cin >> bacanje) {
        korak++;
        if(bacanje == 1) break;

        if(bacanje + pozicija > 10) continue;
        else if(bacanje + pozicija == 10)  {
            kraj = true;
            break;
```

```
        }

        pozicija += bacanje;
        if(pozicija % 2 == 1) pozicija--;
    }

    cout << (kraj ? korak : 0) << endl;

    return 0;
}
```

4. Дефинисати структуру `lubenica` која садржи назив сорте (ниска до 20 слова), тежину у килограмима (позитиван реални број) и цену по килограму (позитиван цео број)

   Немања продаје лубенице и треба му програм за рад са муштеријама. Написати програм који учитава природни број n, а затим податке о n лубеница, а онда име тражене врсте и буџет купца. На стандардни излаз исписати цену најскупље лубенице тражене сорте за коју купац има новца. У случају да не постоји таква лубеница исписати нема. У случају неисправног улаза исписати -1 на стандардни излаз и прекинути програм.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
Улаз: 4 Sunsweet 8.3 100 Fantasy 11.3 80 Sunsweet 9.33 90 Boston 7.6 110 Sunsweet 10000 Излаз: 839	Улаз: 3 Fantasy 9.4 80 Fantasy 11.5 80 Boston 10.5 110 Fantasy 800 Излаз: 752	Улаз: 3 Sunsweet 10 90 Sunsweet 9.5 100 Fantasy 8.3 80 Sunsweet 600 Излаз: nema	Улаз: 3 Sunsweet 10 90 Sunsweet -9.5 100 Излаз: -1

Решење: Дефинишемо тражену структуру. Када учитамо број лубеница направимо вектор са структура са толико елемената. Учитавамо све елементе и проверавамо исправност уноса, ако је неисправан унос исписујемо -1 и прекидамо програм. Након што учитамо тражену лубеницу и буџет прођемо кроз све елементе вектора и проверимо да ли су услови истог назива, цене мање од буџета и цене веће од највеће до тад нађене испуњени. Уколико су сви услови испуњени то је актуелна цена. На крају испишемо цену или нема уколико није пронађена адекватна лубеница.

```
#include <iostream>
#include <vector>
#include <string>
#include <cmath>

using namespace std;

struct lubenica {
```

```
    string naziv;
    double tezina;
    int cenaKg;
};

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

    vector<lubenica> lubenice = vector<lubenica>(n);

    for(int i = 0; i < n; i++) {
        cin >> lubenice[i].naziv >> lubenice[i].tezina >> lubenice[i].cenaKg;

        if(lubenice[i].naziv.length() > 20 || lubenice[i].tezina < 0
        || lubenice[i].cenaKg < 0) {
            cout << -1;
            return 0;
        }
    }

    string naziv; int budzet;
    cin >> naziv >> budzet;
    int cenaNadjena = -1;

    for(int i = 0; i < n; i++) {
        int trCena = floor(lubenice[i].cenaKg * lubenice[i].tezina);
        if(lubenice[i].naziv == naziv && trCena <= budzet && trCena > cenaNadjena)
            cenaNadjena = trCena;
    }

    if(cenaNadjena == -1) cout << "nema";
    else cout << cenaNadjena;

    return 0;
}
```