Увод у програмирање, I смер - Јануар 1

1. Радник у продавници има у каси само новчиће од 1, 2 и 5 динара. Он треба да врати кусур купцу тако што ће му прво дати највећи могући број новчића од 5 динара. Ако тада кусур још увек није отплаћен, радник треба да остатак покрије што већим бројем новчића од 2 динара. Најзад, ако кусур поново није отплаћен, остатак радник исплаћује у новчићима од једног динара.

   Са стандардног улаза се учитава износ који треба вратити као кусур. На стандардни излаз исписати укупан број новчића које радник треба да врати купцу. Претпоставити да је улаз исправно задат.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
`Улаз:`	`Улаз:`	`Улаз:`	`Улаз:`
`24`	`33`	`15`	`1`
`Излаз:`	`Излаз:`	`Излаз:`	`Излаз:`
`6`	`8`	`3`	`1`

   Решење: Да решимо овај задатак потребно нам је целобројно дељење. Број новчића у вредности од пет динара једнак је броју целобројног количника кусура и броја 5 (с обзиром на то да целобројно дељење функционише тако што заокружи количник на доле (нпр. $5.65 \to 5, 3.23 \to 3$)). Од кусура одузмемо производ вредности новчића и њихов број (модул). Број тих новчића саберемо са истим решењем за новчиће од 2 и једног динара и добијемо резултат.

```
#include <iostream>

using namespace std;

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

  int novcici = kusur / 5;
  kusur %= 5;
  novcici += kusur / 2;
  kusur %= 2;
  novcici += kusur;

  cout << novcici << endl;

  return 0;
}
```

2. На такмичењу у кувању, трочлани жири оцењује такмичаре оценама од 1 до 10. Кажемо да је такмичар фаворизован ако му је просек оцена већи од медијалне оцене (средишње у сортираном поретку оцена). За дате оцене такмичара проверити

да ли је такмичар фаворизован и исписати разлику између његове просечне и медијалне оцене.

Са стандардног улаза се учитавају три оцене такмичара (цели бројеви 1–10). На стандардни излаз исписати **da** ако је фаворизован, иначе **ne**, и разлику просека и медијане заокружену на 2 децимале. У случају неисправног улаза исписати -1.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
Улаз: 4 2 7 Излаз: da 0.33	Улаз: 8 7 1 Излаз: ne -1.67	Улаз: 3 3 3 Излаз: ne 0.00	Улаз: 7 4 11 Излаз: -1

Решење: Након што учитамо све оцене такмичара проверимо да ли су у интервалу од 1 до десет, уколико нису испишемо -1 и прекинемо извршавање програма. Уколико је унос био тачан рачунамо просечну оцену тако што збир свих оцена поделимо са 3.0 (овде је важно написати тројку у децималном облику јер би иначе било извршено целобројно дељење јер је унос целобројан). Не морамо да сортирамо оцене да бисмо добили медијалну оцену већ од збира свих оцена одузмемо најмању и највећу и тако добијемо ону у средини. Уколико је просечна оцена већа испишемо да, иначе исписујемо не. Разлику рачунамо стандардно и исписујемо је заокружену на две децимале.

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

using namespace std;

int main() {
  int o1, o2, o3;
  cin >> o1 >> o2 >> o3;

  if(o1 < 1 || o1 > 10 || o2 < 1 || o2 > 10 || o3 < 1 || o3 > 10) {
    cout << -1 << endl;
    return 0;
  }

  double prosecnaOcena = (o1 + o2 + o3) / 3.0;
  int medijalnaOcena = o1 + o2 + o3 - min({o1, o2, o3}) - max({o1, o2, o3});

  if(prosecnaOcena > medijalnaOcena)
    cout << "da ";
  else
    cout << "ne ";

  double razlika = prosecnaOcena - medijalnaOcena;
  cout << showpoint << fixed << setprecision(2) << razlika << endl;
```

```
  return 0;
}
```

3. Написати програм који исписује највећи међу учитаним бројевима који не садржи цифру 3 у запису.

   Са стандардног улаза се уносе цели бројеви до краја улаза. На стандардни излаз исписати највећи од учитаних бројева који не садржи цифру три у запису. Уколико такав број не постоји исписати **nema**. Претпоставити да је улаз исправно задат.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
Улаз:	Улаз:	Улаз:	Улаз:
23	1256	-102	324
25	525	-12	12523
17	-2541	-32	-2345
34	2531	-16	3
53	1802	Излаз:	Излаз:
6413	Излаз:	-12	nema
Излаз:	1802		
25			

Решење: Дефинишемо функцију `sadrziTrojku` да би смо лако проверили да ли број садржи цифру три. Проверавамо број по модулу 10 и уколико је једнак тројци враћамо да број садржи тројку, у сваком кораку број целобројно делимо са 10 након провере. Уколико нисмо наишли ни на једну тројку, а проверили смо цео број, значи он не садржи цифру три. Остатак решења је једноставан, учитавамо све бројеве са стандардног улаза и за сваки проверимо да ли садржи цифру три. Уколико број не садржи цифру три он је највећи уколико је већи од оног до тада провереног. Уколико нисмо доделили ниједан број већи од минималне вредности за `int` исписујемо да нема траженог број, иначе исписујемо број који је резултат.

```
#include <iostream>
#include <climits>

using namespace std;

bool sadrziTrojku(int broj) {
  broj = abs(broj);
  while(broj > 0) {
    if(broj % 10 == 3) return true;
    broj /= 10;
  }

  return false;
}

int main() {
  int broj;
  int najveciBroj = INT_MIN;
```

```
  while(cin >> broj) {
    if(!sadrziTrojku(broj)) najveciBroj = max(najveciBroj, broj);
  }

  if(najveciBroj == INT_MIN) {
    cout << "nema" << endl;
    return 0;
  }

  cout << najveciBroj << endl;
  return 0;
}
```

4. Написати програм који исписује све елементе матрице димензија $n \times m$ који су најмањи у својој колони. Елементе исписати сортирано неопадајуће.

   Са стандардног улаза се учитавају најпре димензије матрице, а затим и њени елементи. На стандардни излаз исписати тражене елементе раздвојене размаком и сортиране неопадајуће. У случају неисправног улаза исписати -1.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
Улаз:			
4 3	Улаз:	Улаз:	Улаз:
5 2 3	3 5	3 3	-3 3
5 1 4	8 4 2 9 22	8 4 3	8 4 3
2 7 0	1 9 2 7 15	4 3 2	4 3 2
5 2 3	2 7 0 9 11	2 7 4	2 7 4
Излаз:	Излаз:	Излаз:	Излаз:
0 1 2	0 1 4 7 11	2 2 3	-1

Решење: Учитавамо димензије матрице и проверимо да ли су димензије у исправном формату. Декларишемо вектор величине m и попуњен је највећим могућим вредностима `int`. Учитавамо матрицу и одма уписујемо вредност у вектор за ту колону уколико је она мања од дотадашње (или на почетку максималне вредности) у тој колони. Сортирамо вектор и испишемо све елементе.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <climits>

using namespace std;

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

  if(n < 1 || m < 1) {
    cout << -1 << endl;
```

```
    return 0;
  }

  vector<int> najmanjeKolone(m, INT_MAX);
  int priv;

  for(int i = 0; i < n; i++) {
    for(int j = 0; j < m; j++) {
      cin >> priv;
      najmanjeKolone[j] = min(najmanjeKolone[j], priv);
    }
  }

  sort(begin(najmanjeKolone), end(najmanjeKolone));

  for(int x : najmanjeKolone) {
    cout << x << " ";
  }

  cout << endl;

  return 0;
}
```

5. Написати програм који проверава колико од унетих n ниски има тачно k различитих карактера у запису.

   Са стандардног улаза се најпре уносе бројеви n и k, а затим и n ниски. На стандардни излаз исписати колико има ниски са тачно k различитих карактера. У случају неисправног улаза исписати -1.

Пример 1	**Пример 2**	**Пример 3**	**Пример 4**
Улаз: 5 3 abc abcd abb abcc abacba Излаз: 3	Улаз: 4 2 abababab 1a11 bbbbb abbbbc Излаз: 2	Улаз: 3 2 abc bac aaa Излаз: 0	Улаз: 3 -2 Излаз: -1

   Решење: Учитамо n и k и проверимо да ли су унети у исправном формату. Дефинишемо скуп карактера и учитавамо све ниске. Када учитамо ниску прођемо кроз сва њена слова и додамо у скуп сва (заправо се додају само она која још нису додата). На крају нам је број различитих слова (карактера) у речи једнак броју елемената скупа и ако је тај број једнак са k инкрементирамо бројач. На крају сваке итерације испразнимо скуп.

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

using namespace std;

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

  if(n < 1 || k < 1) {
    cout << -1 << endl;
    return 0;
  }

  string priv;
  set<char> karakteri;
  int brojac = 0;

  for(int i = 0; i < n; i++) {
    cin >> priv;
    for(int j = 0; j < priv.length(); j++) {
      karakteri.insert(priv[j]);
    }

    if(karakteri.size() == k) brojac++;
    karakteri.clear();
  }

  cout << brojac << endl;

  return 0;
}
```