1. **ShopCart**
   Vytvorte triedu **ShopCart** reprezentujúcu nákupný košík, implementujte metódy **addItem(name, price)**, ktorá do nákupného košíku pridá položku *name* (String) o cene *price* (float), metódu **removeItem(name)**, ktorá odstráni položku z názvom *name*, a metódu printContent(), ktorá vypíše aktuálny zoznam vecí v košíku a ich celkovú hodnotu.

2. **Sensors**
   Vytvorte program, ktorý si načíta názov súboru A. Súbor obsahuje v každom riadku záznam zo senzora - celé číslo. Váš program vypíše na štandardný výstup postupne do riadkov: maximálnu hodnotu v súbore A, minimálnu hodnotu v súbore A, priemer hodnôt v súbore A.

3. **File Statistics**
   Vytvorte program, ktorý načíta názov súboru A. Následne na štandardný výstup vypíše tri riadky - počet riadkov súboru, počet slov súboru a počet znakov v súbore.

4. **Time**
   Definujte triedu **Time** ktorá bude reprezentovať čas. Vhodne nastavte privátne atribúty a implementujte nasledovné metódy:
   - **konštruktor**, ktorý berie dve celé čísla - hodiny a minúty
   - metódu **display()** ktorá vypíše aktuálny čas vo formáte hh:mm
   - metódu **getTime()**, ktorá vráti čas v textovej reprezentácii vo formáte hh:mm
   - metódu **add(h,m)**, ktorá posunie čas reprezentovaný inštanciou o *h* hodín a *m* minút (môžu to byť aj záporné hodnoty)
   - do odovzdaného súboru napíšte aj *globálnu funkciu* **createTime(time, h, m)**, ktorá na základe parametra *time* (inštancie triedy Time) vytvorí novú inštanciu triedy Time, posunutú od *time* o *h* hodín a *m* minút.

5. **Znamky**

Navrhnite triedu **Student** (vytvorte jej privátne atribúty a implementáciu metód), ktoré poskytnú nasledovnú funkčnosť:

- Študent má svoje ID (celé číslo), ID ISICU (String), meno, priezvisko, názov triedy (String - napríklad "IV.A") a emailovú adresu - tieto atribúty sa špecifikujú pri vytváraní inštancie
- Študent má tiež uchovaný zoznam predmetov, ktoré navštevuje.
- Predmety sa dajú pridávať (sú reprezentované Stringom, napríklad "matematika") volaním funkcie **<inštancia_študenta>.addSubject(subject)**, kde subject je String - daný predmet
- predmet môžeme študentovi odobrať volaním funkcie **removeSubject(subject)**, napríklad s1.removeSubject("matematika"), pozor, ak existujú známky z daného predmetu, program musí vyžadovať dodatočné potvrdenie
- k predmetom sa dajú zapisovať známky
- funkciou **addGrade(subject, event_name, points, base)**, kde subject je názov predmetu (jedinečný String), event_name je jedinečný názov udalosti, z ktorej je hodnotenie, napríklad "testik 01", points a base sú desatinné čísla - koľko bodov študent získal a aký bol bodový základ, teda napríklad môžeme zavolať:
  - student1.addGrade("prog", "du1", 3, 10), študent dostal 3/10 za udalosť "du1" z predmetu "prog"
  - ak predmet neexistuje, vypíšte chybu
- implementujte funkciu **removeGrade(event_name)**, ktorá odstráni hodnotenie Študenta z udalosti event_name (String)
- implementujte funkciu **printAllGrades()**, ktorá vypíše prehľad študentových známok - z každého predmetu všetky známky, k nim priemer a vyplývajúce záverečné slovné hodnotenie (výborný / chválitebný / dobrý / dostatočný / nedostatočný)
- implementujte funkciu **printAllGrades(subject)**, ktorá vypíše prehľad známok študenta z daného predmetu, vhodne informujte o neexistujúcom predmete alebo neexistujúcich hodnoteniach za daný predmet

Implementujte funkcie **store(filename)**, ktorá vytvorí súbor *filename.student* obsahujúci všetky informácie o študentovi a funkciu **load(filename)** ktorá zo súboru *filename* načíta dáta o študentovi do inštancie triedy *Student*.

6. Show or hide

Vytvorte program... Program dostane na vstupe celé číslo **x** - počet záznamov, nasledované **x** záznamami (text). Následne program načítava celé čísla - indexy prvkov, ktoré chceme zo zoznamu odstrániť/obnoviť. Program končí načítaním čísla -1, potom vypíše jednotlivé prvky zoznamu pod seba (tie, čo neboli odstránené, resp. boli obnovené).

Pozor!

Načítavané indexy referujú k pôvodnému umiestneniu prvku v zozname a môžu byť duplicitné. Prvý výskyt čísla znamená odstránenie prvku, ďalší jeho obnovu, potom zas odstránenie...

príklad vstupu:

```
7
jablko
hruska
jahoda
salat
kukurica
visna
zemiak
0
1
2
3
4
5
6
1
2
3
3
-1
```

výstup:

```
hruska
jahoda
```

7. Rectangle

Implementujte triedu Rectangle reprezentujúcu Obdĺžnik. Vytvorte metódy:

- **konštruktor**, ktorý vytvorí obdĺžnik so stranami a,b a polohou ľavého horného rohu [x,y]
- **setDimensions(a,b)**, ktorá zmení rozmery obdĺžnika
- **getDimensions()**, ktorá vráti tuple - dvojicu obsahujúcu dĺžky strán obdĺžnika
- **isSquare()**, ktorá vráti boolean, na základe toho, či je obdĺžnik štvorcom (na základe rozmerov)
- **getArea()**, ktorá vráti číslo - veľkosť plochy obdĺžnika
- **getCircumference()**, ktorá vráti číslo - dĺžku obvodu obdĺžnika
- **getPosition()**, ktorá vráti dvojicu obsahujúcu súradnice [x,y] ľavého horného rohu obdĺžnika
- **move(deltaX, deltaY)**, ktorá posunie obdĺžnik na ploxe o deltaX bodov v osi X a o deltaY bodov v osi Y
- **moveTo(x,y)**, ktorá presunie obdĺžnik na súradnice [x,y]
- **contains(x,y)**, ktorá vráti boolean podľa toho, či je bod na súradniciach [x,y] v obdĺžniku (počíta sa aj na obvode)
- **makeSquare()** ktorá zmení obdĺžnik na štvorec, tak, že dlhšiu stranu skráti na dĺžku menšej strany

8. LIFO

Vytvorte triedu **Zasobnik** reprezentujúcu zásobník typu LIFO (last in, first out) implementujte metódy na pridávanie - **push(item)** a výber prvkov zásobníku - **pop()**. Prvkom zásobníku môže byť ľubovoľná hodnota. Implementujte tiež metódu **display()**, ktorá zobrazí aktuálny stav zásobníka (zobrazí všetky prvky). isEmpty( ) a len( )

9. FIFO

Vytvorte triedu **Fronta** reprezentujúcu frontu (zásobník typu FIFO - first in, first out), implementujte metódy na pridávanie - **push(item)** a výber prvkov zásobníku - **pop()**. Prvkom zásobníku môže byť ľubovoľná hodnota. Implementujte tiež metódu **display()**, ktorá zobrazí aktuálny stav zásobníka (zobrazí všetky prvky).