

**Obrni prvih k elementov vrste**

RAČUNALNIŠTVO 1

|  |  |
| --- | --- |
|  | **Avtor naloge:** |
|  | Gal Zakrajšek |

Ljubljana, 2019

Kazalo

[Vprašanje 2](#_Toc23451772)

[Dodatna pravila 2](#_Toc23451773)

[Ideja rešitve 2](#_Toc23451774)

[Robni primeri 5](#_Toc23451775)

[Negativen k 5](#_Toc23451776)

[K večji od dolžine vrste 5](#_Toc23451777)

[Prazna vrsta in k večji od 0 5](#_Toc23451778)

[Izogib napakam in nesmiselnim podatkom 5](#_Toc23451779)

[Viri 6](#_Toc23451780)

# Problem

Na naslednjih straneh bo prikazan eden iz med načinov, kako obrniti prvih elementov v neki vrsti. Program bo za parametre dobil neko vrsto in pa število , vrniti pa mora isto vrsto, v kateri je prvih elementov v obratnem vrstnem redu.

## Dodatna pravila

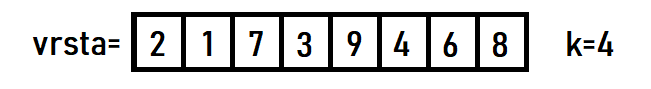
Pri nalogi imamo tudi eno dodatno pravilo in sicer na vrsti lahko uporabljamo samo naslednje metode:

* **vstavi(x)**: Vstavi element x na konec vrste
* **izbrisi()**: Odstrani element iz vrste
* **velikost()**: Vrne število elementov v vrsti
* **prvi()**: Vrne prvi element v vrsti

Sedaj, ko poznamo našo nalogo in pravila, se lahko lotimo ideje.

# Primeri

Oglejmo si nekaj primerov. Vrsta je narisana tako, da je njen začetek na levi.



# 

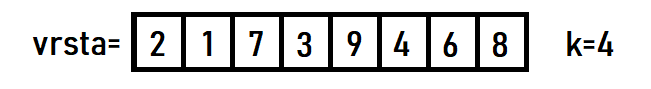
# Tu bi dal potem še denimo primer, ko je k enak velikosti vrste (ko obrnemo celo vrsto) in primer, ko je k == 1 (ko vrsta ostane enak).

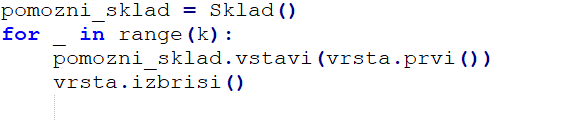
# Ideja rešitve

Tu res napišemo idejo. Najprej povemo, v čem je težava --- k elementov moramo najprej odstraniti iz vrste, potem pa jih dodati nazaj, a v obratnem vrstnem redu in na začetek.

Za obračanje bomo poskrbeli s tem, da bomo prvih k elementov najprej shranili v sklad in jih nato dodali na konec. Da bodo prišli na začetek pa bomo poskrbeli s tem, da bomo preložili preostanek elementov, pri čemer bomo morali vedeti, koliko jih je. Za to pa bo poskrbela funkcija velikost.

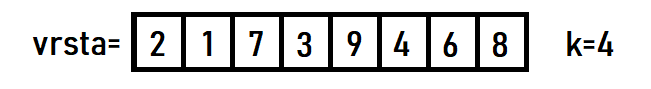
Oglejmo si postopek na primeru. Denimo, da imamo

Kot smo rekli, najprej elementov iz vrste vstavimo v pomožni sklad. To naredimo z zanko, ki se izvrši krat in pri tem vsakič iz vrste vzame prvi element in ga vstavi v sklad.

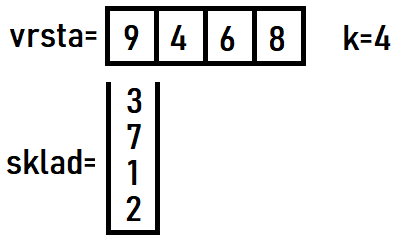


Slika 3: koda prve zanke v programu

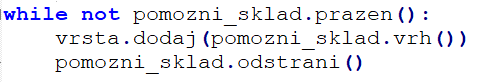
Iz



torej dobimo

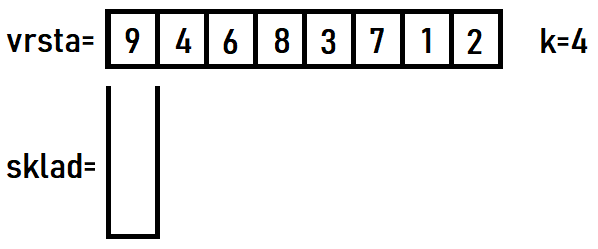


Tako imamo sedaj vrsto, v kateri so samo še elementi, katerih vrstni red mora ostati enak kakor na začetku in pa sklad, v katerem so vsi elementi, katerih vrstni red je potrebno obrniti.   
Sedaj iz sklada začnemo jemati podatke in jih dodajati na konec vrste. Ko je sklad prazen, dobimo vrsto, ki ima na začetku vse elemente katere smo pustili v vrsti zatem pa vse elemente, ki pa imajo zdaj obrnjen vrstni red. To naredimo tako, da z zanko delamo toliko časa dokler imamo še kakšne podatke v skladu. V vsaki iteraciji pa vzamemo podatek iz sklada in ga vstavimo v vrsto.



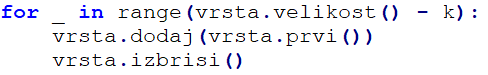
Slika 5:koda druge zanke v programu

Po izvršitvi druge zanke bo stanje podatkov naslednje:



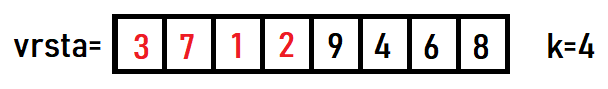
Slika 6: stanje podatkov po drugi zanki

Ugotovimo, da imamo zdaj na levem delu vrste elemente, katere imajo začetni vrstni red. Na desni strani pa elemente, katerim smo obrnili vrstni red. Da bomo prišli do želene rešitve, moramo premakniti celoten levi del začetno urejenih podatkov na konec vrste. Pri določanju števila elementov, ki jih je zdaj potrebno premakniti iz začetka na konec vrste, si pomagamo z metodo velikost. Če je vseh elementov in smo že obrnili elementov, moramo iz začetka na konec premakniti elementov. To naredimo z zanko, ki se izvrši krat in pri tem iz začetka vrste vzame element in ga prestavi na konec.



Slika 7: koda tretje zanke v programu

Ko se izvrši še zadnja zanka, pridemo do pravilne postavitve elementov in s tem do končne rešitve, pri kateri ima prvih elementov v vrsti obrnjen vrstni red.



Slika 8: stanje po končanem programu

# Koda rešitve

\*\*\* NAVEDEMO KODO \*\*\*

# Morebitne težave

Pri funkciji moramo biti pozorni tudi na primere, ko podani parametri niso usklajeni. V našem primeru imamo 3 primere, pri katerih moramo paziti, kaj bo program storil.

## Negativen k

V primeru, da programu podamo neko vrsto celih števil in pa zraven negativen , se bo naš program vseeno izvedel do konca brez vmesnih napak, vendar pa bo rešitev napačna oziroma nesmiselna. Če se spomnimo programa in zank, ki jih imamo, ugotovimo, da se prvi dve zanki sploh ne bosta izvršili. Zato bo program takoj preskočil na zadnjo zanko. Ta zanka se bo v tem primeru izvršila krat torej krat. Tako bo samo elementov premaknila iz začetka na konec. Posledično bo naš program vrnil isto vrsto samo zamaknjeno za mest.

## K večji od dolžine vrste

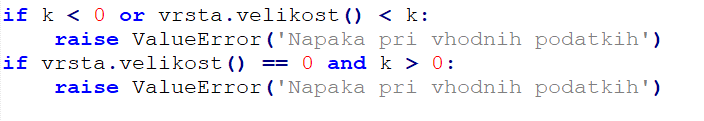
V tem primeru imamo vrsto, katere dolžina je manjša od podanega števila k. Pri takih podatkih bo program vrnil napako že pri prvi zanki, saj bo poskušal iz premakniti k elementov iz podane vrste. To pa je nemogoče, saj vrsta nima toliko elementov.

## Prazna vrsta in k večji od 0

V primeru, da imamo podano prazno vrsto in k, ki je enak 0, bo program deloval pravilno, saj se ne bo izvršila nobena zanka in bo program vrnil nazaj prazno vrsto. Kar je teoretično pravilna rešitev. Problem pa nastane, če je dana prazna vrsto in k > 0. Takrat se zgodi enako kot če je k večji od dolžine vrste.

## Dopolnitev kode

Zgoraj navedenim napakam oziroma nesmiselnemu delovanju programa se lahko izognemo tako, da na začetku preverimo, ali podatki ustrezajo, oziroma ali so smiselni. če temu ni tako, sprožimo napako vrste ValueError.



Slika 9: Pogojni stavek, za preverjanje ustreznosti vhodnih parametrov

Dopolnjena koda!!!

# Razširitev problema

Denimo, da funkcija velikost ne bi bila dana. Kako pa bi potem rešili naš problem?

# Viri

Sharma R. *Reversing the first K elements of a Queue | GeeksforGeeks (b. d.).* Pridobljeno s[https://www.geeksforgeeks.org/reversing- first-k-elements-queue/](https://www.geeksforgeeks.org/reversing-first-k-elements-queue/) DATUM DOSTOPA????