

**SEMINARSKA NALOGA**

PROGRAMIRANJE 1

PRAKTIČNA MATEMATIKA

**Avtor naloge:**

Aljaž Marn

Ljubljana, 2024/25

|  |  |  |  |
| --- | --- | --- | --- |
|  | Ime naloge | Aktivna povezava | Število težavnostnih točk |
| 1. del | 4 toughts | <https://open.kattis.com/problems/4thought> | 3.0T |
|  | A prize no one can win | <https://open.kattis.com/problems/aprizenoonecanwin> | 3.0T |
|  | Adding words | <https://open.kattis.com/problems/addingwords> | 4.1T |
|  | Skupaj točk |  | 11.1T |
| 1 DEL | Ocena |  | 11 T |
|  |  |  |  |
| 2. del | Alice in the digital world | <https://open.kattis.com/problems/alicedigital> | 4.6 T |
|  | Bank queue | <https://open.kattis.com/problems/bank> | 2.8 T |
|  | Robotopia | <https://open.kattis.com/problems/robotopia> | 5.4 T |
|  | Skupaj točk |  | 12.7 T |
| SKUPAJ |  | **11T + predvidoma 13T** | 24T |

Iz 1. dela sem popravil nalogo:

* **A prize no one can win** <https://open.kattis.com/problems/aprizenoonecanwin> - dodal sem utemeljitev zakaj pridemo do rešitve tako, da seštevamo zaporedna elementa v urejenem seznamu.

Popravki nadomeščajo staro besedilo.

# Izjava

*Aljaž Marn izjavljam, da sem seminarsko nalogo opravil samostojno in da sem njen avtor. To pomeni, da sem v poročilu točno označil vse tiste dele kode, ki sem jih povzel iz virov in vire navedla. Prav tako vem, da izjava o avtorstvu pomeni, da znam razložiti vsako podrobnost kode, ki jo oddajam. Zavedam se, da v primeru, če izjava prvega stavka ni resnična, kršim disciplinska pravila*

**POROČILO 1. del**

# 4 toughts

Naloga je na spletni strani <https://open.kattis.com/problems/4thought> in je vredna 3.0 točke.

## BESEDILO NALOGE

Napišite program, ki bo ob podanem celoštevilskem vhodu n ustvaril matematični izraz, katerega rešitev je n. Rešitev mora uporabiti natanko štiri števke 4 in natanko tri binarne operacije, izbrane iz množice {∗,+,−,/}. Število 4 je EDINO število, ki ga lahko uporabite. Ni dovoljeno združevati števk 4 za ustvarjanje drugih števil, kot sta 44 ali 444.

Na primer, za n=0 je rešitev 4∗4−4∗4=0. Za n=7 je rešitev 4+4−4/4=7. Deljenje se obravnava kot celoštevilsko deljenje z okrajšavo, tako da je 1/4 enako 0 (namesto 0,25). Upoštevajte običajno prednost operacij, tako da je 4+4∗4=20, ne 32. Nimajo vsi celoštevilski vhodi rešitve z uporabo štirih števk 4 z omenjenimi omejitvami (razmislite o n=11).

**Vhodni podatki:**

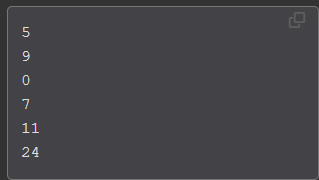
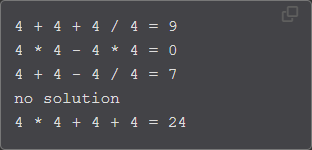
V prvi vrstici je podano celoštevilsko število m. To število je med 1 in 1000. V naslednjih m vrstic, pa je podano celoštevilsko število n. To število pa je med -1000000 in 1000000.

**Izhodni podatki:**

Program izpiše m vrstic. V vsaki je račun, katere rešitev je n. Izhod mora biti formatiran takole (naj znak $ označuje, katero koli izmed dovoljenih operacij):

4 $ 4 $ 4 $ 4 = n

Če tak račun ne obstaja naj program izpiše: no solution.

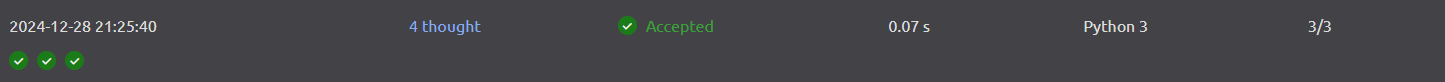
Primer vhodnih podatkov: Primer izhodnih podatkov:

## OPIS PROBLEMA IN IDEJA REŠEVANJA

Prva vrstica vhodnih podatkov nam pove količino testnih primerov. Preostanek pa so rešitve računa katerega moramo sestaviti.

Preberimo najprej prvo vrstico. Potem prebiramo vrstice eno po eno in sproti izpisujemo račune. Pregledujemo vse možne kombinacije operatorjev. Vsak izraz tudi izračunamo in če je izračun enak vhodu, ga izpišemo. Pri računanju moramo paziti na pravilni vrstni red operatorjev, zato najprej izračunamo operatorje, ki imajo »prednost«, tj. najprej množimo in delimo, potem pa seštevamo in odštevamo.

## DOKAZ, DA JE NALOGA REŠENA



# A prize no one can win

Naloga je na spletni strani <https://open.kattis.com/problems/aprizenoonecanwin> in je vredna 3 točke.



## BESEDILO NALOGE

Po prazničnem odprtju vaše nove trgovine, Butična trgovina za alternativno paramedicino in Cwakhsahlvereigh, z razočaranjem ugotovite, da ne dosegate toliko prodaje, kot ste upali. Da bi to popravili, se odločite za posebno ponudbo: označili boste določeno podmnožico od n izdelkov v vaši trgovini kot sodelujoče v ponudbi, in če ljudje kupijo natančno dva od teh izdelkov, pri čemer je strošek teh izdelkov \*strogo\* večji od X evrov, jim boste podarili brezplačen samorogov rog!

Ker ste pred kratkim ugotovili, da so vsi vaši samorogovi rogovi pravzaprav narvaljevi okli, se odločite prelisičiti ponudbo tako, da boste sodelujoče izdelke izbrali na način, da nihče tako ali tako ne bo mogel zaslužiti roga.

Da ne bi nihče postal sumničav, želite označiti čim več izdelkov kot sodelujočih v ponudbi.

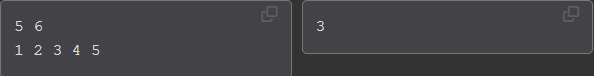
Vhodni podatki:

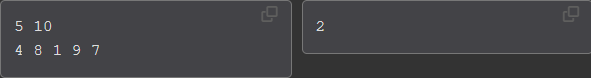
V prvi vrstici, sta dve števili. Prva označuje število izdelkov (pozitivno število manjše od 100000), druga pa ceno, ki jo mora kupec preseči, da dobi samorogov rog (pozitivno število manjše od bilijona).

V naslednji vrstici pa so cene izdelkov. To so pozitivna števila, ki so manjša od bilijona.

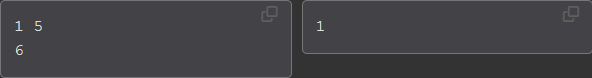
Izhodni podatki:

Število izdelkov, ki jih lahko označimo, da sodelujejo v ponudbi, s tem, da nihče ne more dobiti roga.

Primer vhodnih podatkov 1: Primer izhodnih podatkov 1: 

Primer vhodnih podatkov 2: Primer izhodnih podatkov 2:

Primer vhodnih podatkov 3: Primer izhodnih podatkov 3:

Primer vhodnih podatkov 4: Primer izhodnih podatkov 4: 

## OPIS PROBLEMA IN IDEJA REŠEVANJA

Vhodne podatke sestavljata dve vrstici. V prvi vrstici sta dve število, število izdelkov in cena, ki jo mora kupec preseči. V drugi vrstici pa so cene izdelkov, ki se prodajajo v trgovini.

Preberemo število n in izdelkov in ceno x. Potem preberemo cene izdelkov in jih za lažje obravnavanje uredimo po velikosti. Potem pogledamo dva zaporedna elementa in ju seštejemo. Če je seštevek manjši od x, vemo, da ta element lahko označimo za ponudbo. To lahko naredimo zato, ker je seznam urejen, saj če sta zaporedna elementa pod x, potem so tudi vse kombinacije z manjšimi elementi.

## DOKAZ, DA JE NALOGA REŠENA



# Adding Words

Naloga je na spletni strani <https://open.kattis.com/problems/addingwords> in je vredna 4.1 točke.



## BESEDILO NALOGE

Psihologi na Univerzi Wassamatta verjamejo, da ljudje lažje upravljajo z besedami kot s številkami. Zato so zasnovali eksperimente, da bi ugotovili, ali to drži. V zanimivem preobratu se eden od njihovih eksperimentov ukvarja z uporabo besed, ki predstavljajo številke. Namesto seštevanja številk želijo seštevati besede. Vi ste raziskovalni programer na projektu in vaša naloga je napisati program, ki prikazuje to sposobnost.

Vhodni podatki:  
Vhod je zaporedje do 2000 ukazov. Vsak ukaz je v svoji vrstici. Ukazi se zaključijo, ko se prebere celotna datoteka. Vsak ukaz je ali definicija ali račun ali pa izbris. Vsi elementi znotraj ukaza so ločeni z enojnim presledkom.

Definicija ima obliko *def x y,* kjer je x ime spremenljivke, y pa celo število, ki je večje od -1000 in manjše od 1000. Dve spremenljivki, ne moreta biti hkrati definirani z isto številčno vrednostjo. Če je bila x že prej definirana, nova definicija izbriše njeno staro definicijo. Imena spremenljivk so lahko dolga do 30 malih črk, vsaka črka med a in z.

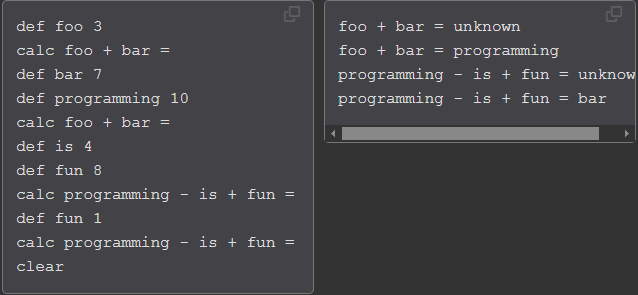
Ukaz za račun se začne z besedo *calc* sledijo pa ji ena do 15 spremenljivk, ločenih z operatorji za seštevanje in odštevanje. Konec ukaza, je enačaj. Na primer: calc foo + bar – car +

Ukaz *clear* naroči programu naj pozabi vse svoje definicije.

Izhodni podatki:

Vaš program ne sme proizvesti nobenega izpisa za definicije, za izračune pa mora proizvesti vrednost izračuna. Če za rezultat ne obstaja beseda ali če katera od besed v izračunu ni bila definirana, mora biti rezultat izračuna »unknown«. Beseda »unknown« se nikoli ne uporablja kot spremenljivka v vhodu.

Primer vhodnih podatkov: Primer izhodnih podatkov:



## OPIS PROBLEMA IN IDEJA REŠEVANJA

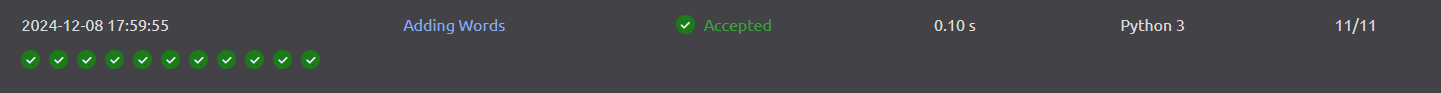
Vhodni podatek je vrstica, ki lahko ali definira novo spremenljivko, računa, ali pa pobriše vse obstoječe definicije. Vrstice je treba brati dokler ne pridemo do konca datoteke.

Če je prva beseda v ukazu »*def*«, si ime spremenljivke in njeno vrednost nekam zapomnimo.

Če je prva beseda v ukazu »*calc*«, vse besede po njihovih definicijah pretvorimo v številke, če kakšna beseda ni definirano izpišemo ukaz brez »*calc*« in na koncu še »*unknown*«. Te številke potem shranjujemo v seznam. Če moramo številko odšteti jo pomnožimo z -1. Na koncu potem to seštejemo in pogledamo ali kakšna izmed spremenljivk ustreza našemu rezultatu, če ne izpišemo ukaz brez »*calc*« in na koncu še »*unknown*«. Če pa rezultat ustreza spremenljivki, izpišemo ukaz brez »*calc*« ter to spremenljivko.

Če je prva in edina beseda v vrstici »*clear«* izbrišemo vse definicije.

## DOKAZ, DA JE NALOGA REŠENA



**POROČILO 2. del**

# Alice in the digital world

Naloga je na spletni strani <https://open.kattis.com/problems/alicedigital> in je vredna 4.6T.

## BESEDILO NALOGE

Po vrnitvi iz Čudežne dežele mora Alica izboljšati svoje znanstvene spretnosti v današnjem digitalnem svetu. Alica se odloči, da se bo udeležila regionalnega tekmovanja ACM - ICPC Azija Nha Trang 2016, da oceni svoje dejanske sposobnosti. Njena najljubša naloga na tekmovanju je opisana spodaj.

Dan je seznam celih števil A = a1, a2, …, an. Pod seznam Aji od A je zaporedje zaporednih elementov v seznamu A. Aji = ai, ai+1, …, aj ; pri čimer je 1 ≤ i ≤ j ≤ n. Teža tega podseznama ja vsota vseh njenih elementov.

Dano je celo število ***m***, tvoja naloga pa je najti pod seznam z največjo težo, ki vsebuje samo eno število ***m*** kot najmanjši element. Lahko predpostavimo, da polje vedno vsebuje vsaj en element z vrednostjo ***m***.

**Vhod**

Vhod obsega več podatkovnih nizov. Prva vrstica vsebuje število podatkovnih nizov, ki je pozitivno število in ni večje od 20. Naslednje vrstice opisujejo posamezne podatkovne nize.

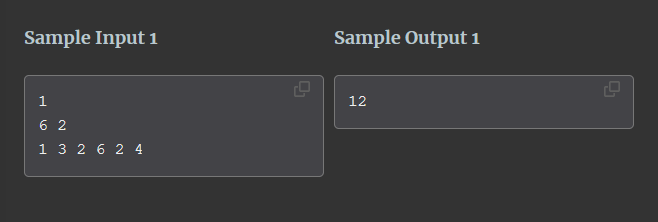
Vsak podatkovni niz je opisan z naslednjimi vrsticami:

* Prva vrstica vsebuje dve pozitivni celi števili n in m, kjer je n ≤ 100000 in m ≤ 26.
* Druga vrstica vsebuje n pozitivnih celih števil, vsak z vrednostjo največ 26.

**Izhod**

Za vsak podatkovni niz izpišite največjo težo.

PRIMER VHODNIH PODATKOV PRIMER IZHODNIH PODATKOV

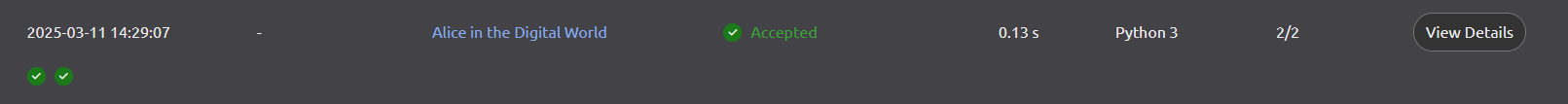


## OPIS PROBLEMA IN IDEJA REŠITVE

Iz seznama moramo najti podseznam zaporednih elementov z največjim seštevkom, ki vsebuje natanko en element m, ki je tudi najmanjši element.

Najprej preberemo število primerov, potem pa vsakič preberemo n (velikost seznama) in m (najmanjši element), ter seznam. To rešimo s funkcijo, ki gre skozi seznam vsak element posebej. Ko najde element m, gre od tega elementa levo in desno dokler ali ne najde manjšega elementa od m ali pa elementa m. Potem dobimo indekse i in j, od kje do kje je pod seznam. To potem seštejemo skupaj in si shranimo največjo težo takega podseznama.

## DOKAZ, DA JE NALOGA REŠENA



## DATOTEKA

alicedigital.py

# Robotopia

Naloga je na spletni strani <https://open.kattis.com/problems/robotopia> in je vredna 5.4T.

## BESEDILO NALOGE

V čudoviti deželi Robotopiji je življenje boljše kot kadarkoli prej. To je zahvaljujoč robotom, ki trdo delajo skupaj. Različni tipi robotov prihajajo v različnih oblikah, torej z različnim številom rok in nog. Čez dan, ko delajo, so roboti združeni v skupine, da jim lahko dodelijo različna opravila. Toda ponoči je vaša naloga, da preštejete, koliko robotov vsake vrste je v vsaki skupini. Roboti so nagajivi in polni energije; ne marajo stati mirno in jih je težko prešteti. Po mnogih izčrpavajočih poskusih vam končno uspe prešteti, koliko rok in nog je skupaj.

Vemo, da vsaka skupina vsebuje točno dve različni vrsti robotov, pri čemer je v vsaki skupini vsaj en robot vsake vrste. Prav tako vemo, koliko rok in nog ima vsaka vrsta robota. Ali lahko ugotovite, koliko robotov vsake vrste je v vsaki skupini?

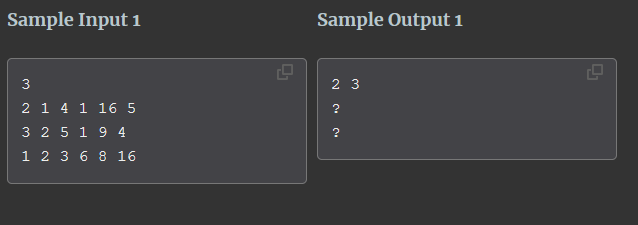
**Vhod**

Prva vrstica vsebuje celo število **t**. Vsaka naslednja vrstica vsebuje šest celih števil: **l1, a1, l2, a2, L, A**, kjer sta **l1** in **a1** število nog in rok (ustrezno) za prvo vrsto robota, **l2** in **a2** pa za drugo vrsto, **L** in **A** pa sta skupno število opazovanih nog in rok. Vse vrednosti so v območju od 1 do 1000.

**Izhod**

Za vsak podatkovni niz izpišite dve pozitivni celi števili, ki označujeta število robotov vsake vrste. Najprej izpišite število za prvo vrsto, nato za drugo. Če ni rešitve ali če je več rešitev, izpišite "?" (vprašaj).

PRIMER VHODNIH PODATKOV PRIMER IZHODNIH PODATKOV



## OPIS PROBLEMA IN IDEJA REŠITVE

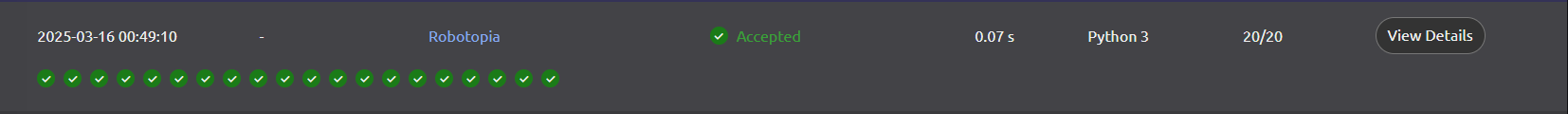
Izračunati moramo koliko robotov z a1 rok in l1 nog in koliko robotov z a2 rok in l2 rok je v seštevku vseh rok in nog.

Najprej preberemo število primerov. Potem beremo l1, a1, l2, a2, lt in at. Število rok in nog prvega modela robota, število rok in nog drugega modela robota in koliko rok in nog je bilo preštetih skupaj.

Ker lahko nastopajo velike številke si izberemo zgornjo mejo (Zm) – do kam gremo preverjati koliko je robotov enega tipa in koliko drugega. S števcem for zanke predstavimo število robotov prvega tipa. Za števec, ki se premika od 1 do Zm izračunamo ali se število nog izide, če se potem izračunamo koliko robotov drugega tipa je. To potem zapakiramo v tuple in dodamo v seznam rešitev.

Če je na koncu izvajanja programa v funkciji samo en rezultat, ga vrnemo. Drugače pa vrnemo (-1, -1), kar simbolizira, da ali je rešitev več ali pa da rešitve ni – vsekakor napačen rezultat in izpišemo '?'.

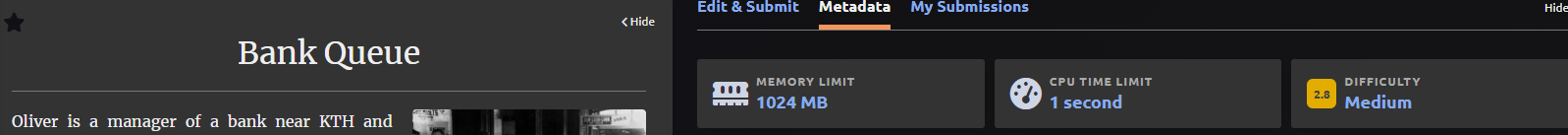
## DOKAZ, DA JE NALOGA REŠENA



## DATOTEKA

robotopia.py

# Bank Queue

Naloga je na spletni strani <https://open.kattis.com/problems/bank> in je vredna 2.8 točke.

## BESEDILO NALOGE

Oliver je vodja banke blizu KTH in želi kmalu zapreti banko. V vrsti stoji veliko ljudi, ki želijo položiti gotovino na svoje račune, potem ko so slišali, da je banka zvišala obrestne mere za 42% (iz 0.01% na 0.0142% na leto). Vendar je ljudi preveč in odprto je le eno okence, ki lahko postreže eno osebo na minuto. Oliver je pohlepen in bi rad zbral ljudi v vrsti tako, da bi bil skupni znesek gotovine, ki jo bodo ljudje položili čim večji, saj bo ta denar potem čez noč delal za banko.

Obstaja pa težava. Nekateri ljudje nimajo časa čakati do zaprtja banke, saj morajo drugam, zato morajo biti postreženi do določenega časa, sicer odidejo. Oliver je tudi izključil senzor pri vhodnih vratih, tako da nihče več ne more vstopiti, ker je avla že prepolna.

Pomagaj Oliverju izračunati, koliko gotovine lahko dobi od ljudi, ki trenutno stojijo v vrsti, preden banka zapre – ob predpostavki, da lahko na minuto postreže največ eno osebo.

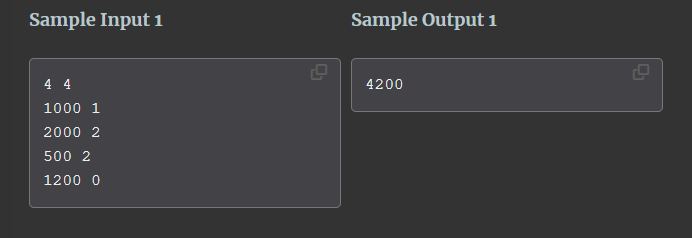
**Vhodni podatki:**

Prva vrstica vhodnih podatkov vsebuje dve celi števili **n** (1 ≤ n ≤ 10000) in **t** (1 ≤ t ≤ 47), kjer je **n** število ljudi v vrsti in **t** čas v minutah do zaprtja banke. Sledi **n** vrstic, vsaka z dvema celoštevilskima vrednostma **c** in **d**, kjer **c** pomeni količino gotovine v švedskih kronah, ki jo ima oseba,  
in **d** pomeni čas v minutah od tega trenutka naprej, po katerem oseba odide, če ni postrežena.  
Za vsako osebo velja, da traja ena minuta, da je postrežena, in da jo je treba začeti streči najkasneje pri času **d**. Velja lahko domnevati, da so **1 ≤ n ≤ 10⁴** in **0 ≤ d < t ≤ 10⁴**.

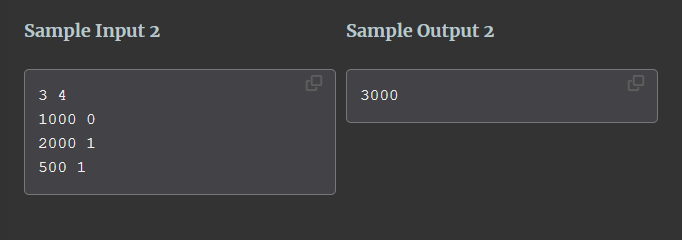
**Izhodni podatki:**

Izpiši eno vrstico z največjo možno količino denarja, ki jo lahko dobiš od ljudi v vrsti, preden se banka zapre.

PRIMER VHODNIH PODATKOV 1 PRIMER IZHODNIH PODATKOV 1



PRIMER VHODNIH PODATKOV 2 PRIMER IZHODNIH PODATKOV 2



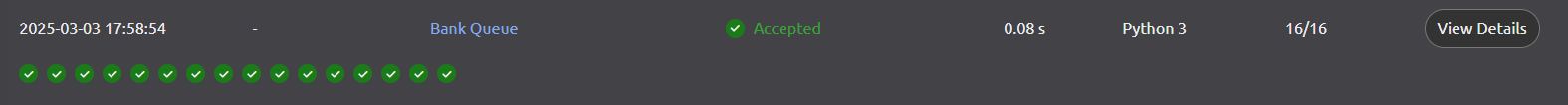
## OPIS PROBLEMA IN IDEJA REŠITVE

Izračunati moramo kolikšna je maksimalna vrednost denarja, ki ga lahko položimo v banko pred zaprtjem.

Najprej preberemo n (število ljudi) in t (koliko je časa je banka še odprta). Potem preberemo podatke vsakega človeka (koliko denarja ima in do katere minute čaka. Potem pa za vsako minuto (od zadnje minute do prve minute) iščemo osebo, ki še ni bila postrežena, lahko počaka vsaj do trenutne minute in ima največ denarja. Če tako osebo najdemo, njen denar dodamo k rezultatu in jo označimo kot postreženo.

Zakaj začnemo pri zadnji minuti in gremo proti prvi? Zato, da si najprej »rezerviramo«, tiste ljudi, ki imajo največ denarja in lahko počakajo najdlje, da jih nebi slučajno uporabili prej.

## DOKAZ, DA JE NALOGA REŠENA



## DATOTEKA

bank.py