Oinarrizko Prog. - 6. laborategia. Zerrendak eta matrizeak

Izena:_____**Data**:_____

OHARRAK:

1. Proposatutako ariketetakoren batek ez badu proba programarik edo txantiloirik, zuek zerotik sortu behar duzuela esan nahi du. Gainera, proba kasu batzuk eskaintzeak ez du esan nahi guztiak daudenik. Falta direnak gehitu beharko dituzue.

2. Laborategi honetan ez duzue informerik bete behar. Soilik, jatorri fitxategiak (.adb eta .py fitxategiak) .zip batean konprimituta igo beharko dituzue eGelara. Fitxategiaren izena izendatze arauak bete beharko ditu (adibidez, JSaez_PEtxebarria_lab6.zip).

3. Ariketak zuzenak direla aurreikusten da, hau da, ez dituzte konpilazio errorerik, eta beraien funtzionamendua zuzena da. Horren arabera, ariketek ez dut positiboan zenbatzen ondo baldin badaude, baina okerrak egitekotan zigortu egingo da. Behin soluzioa zuzena dela, honakoa da ebaluatuko dena (puntuak emango dizkizuena):

   a) Proba kasuak: Proba kasu guztiak aurreikusi dira, orokorrenetatik, kritikoenetara? Batzuk ematen dira, baina beste asko falta dira.

   b) Efizientzia: beharrezkoa den kasutan "salatariak" (txibatoak) erabiltzen dira? Beharrezkoak ez diren esleipenak daude? Exekutatu beharko ez liratekeen baldintzarik dago? Beharrezkoak diren parametro zein aldagaiak bakarrik definitzen dira?

   c) Argitasuna: Kodea tabulatua dago? Aldagaien izenek kodea ulertzen laguntzen dute? *return* bakarra dago funtzioaren bukaeran? ...

4. Datu moten ezaugarriak, **bektoreak.ads** fitxategian topatuko dituzue. Gogoratu, taulek/arrayek/bektoreek ezin dutela luzera aldakorra izan. Baina taula bat erregistro baten barruan sartzen badugu non erregistroak 2 eremu izango dituen, alegia bektorea, eta luzera markatzeko erabiliko dugun *integer* eremua, orduan *"luzera aldakorreko"* taula bat simula dezakegu. Honela, elementu berri bat sartu nahi dugun bakoitzean luzerari bat gehituko diogu eta elementu bat kendu nahi dugun bakoitzean luzerari bat kenduko diogu.

1. Ariketa: Bektore batean bilaketa

Ariketa hau **ADAz** zein **Pythonez** ebatzi behar da.
Fitxategia (ADA): bektoreak.ads (ez da moldatu behar).
Txantiloiak (ADA): bektorean_dago.adb eta proba_bektorean_dago.adb
Txantiloia (Python): bektorean_dago.py

Zenbaki osoko bat eta N osoko dituen bektore bat emanda (N>0 eta elementuak EZ daude ordenatuta), azpiprograma bat egin osoko hori bektorean dagoen ala ez adierazten duena.

2. Ariketa: Bektore ordenatu batean bilaketa

Ariketa hau **ADAz** zein **Pythonez** ebatzi behar da.
Fitxategia (ADA): bektoreak.ads (ez da moldatu behar).
Txantiloiak (ADA): bektore_ordenatuan_dago.adb eta proba_bektore_ordenatuan_dago.adb
Txantiloia (Python): bektore_ordenatuan_dago.py

Zenbaki osoko bat eta N osoko ordenatuak dituen bektore bat emanda (N>0), azpiprograma bat egin osoko hori bektorean dagoen ala ez adierazten duena.

OHARRA: Soluzioa eraginkorra izatea ezinbestekoa da. Zehazki, ez da elementuaren bilaketarekin jarraituko objetiboki ez dagoela dakigunean ordenatuta dagoelako.

3. Ariketa: Bektore batean posizioaren bilaketa

Ariketa hau **ADAz** zein **Pythonez** ebatzi behar da.
Fitxategia (ADA): bektoreak.ads (ez da moldatu behar).
Txantiloiak (ADA): posizioan_dago.adb eta proba_posizioan_dago.adb
Txantiloia (Python): posizioan_dago.py

N elementu dituen osoko-bektore bat eta zenbaki bat emanda (N>0), zenbaki hori zerrendan balego, zein posiziotan dagoen itzuliko duen algoritmoa espezifikatu eta inplementatu**.** Zenbakia hainbat aldiz agertzen bada, nahi duzuen posizioa itzuli baina ez balego, itzuli -1 posizioa**.**

4. Ariketa: Posizio bat eskuinera mugitu

Ariketa hau **ADAz** zein **Pythonez** ebatzi behar da.
Fitxategia (ADA): bektoreak.ads (ez da moldatu behar).
Txantiloiak (ADA): bektorea_idatzi.adb, eskuinera_mugitu.adb eta proba_eskuinera_mugitu.adb
Txantiloia (Python): eskuinera_mugitu.py

N elementuz osatutako osoko-bektore bat emanda (N>0), osagai guztiak posizio bat eskuinera mugitzen dituen algoritmoa espezifikatu eta inplementatu.

Sarrera:

1	2	3	4	5	6	7	8	9	10
34	67	45	23	78	12	40	55	24	89

Irteera:

1	2	3	4	5	6	7	8	9	10
89	34	67	45	23	78	12	40	55	24

5. Ariketa: Ezabatu hirugarren elementua bektore ez ordenatuan

Ariketa hau **ADAz** bakarrik ebatzi behar da.
Fitxategia (ADA): bektoreak.ads eta idatzi_zerrenda.adb (ez dira moldatu behar).
Txantiloiak (ADA): ezabatu_hirugarren_elementua.adb eta proba_ezabatu_hirugarren_elementua.adb

Gehienez N elementu izango dituen osoko-bektore bat emanda (N>0), hirugarren elementua ezabatzen duen azpiprograma inplementatu (baldin badago behintzat). Behar bada bektorea ez da guztiz beteta egongo eta hori kudeatzeko erregistro baten barruan egongo da non erregistroak bektoreaz gain kopuru eremu bat ere izango duen (ikusi *bektoreak.ads*). Bektorean hiru elementu baina gutxiago badaude, orduan ez da eze egingo.

Bektorearen elementuak ez daude ordenatuta (ez ezabaketaren aurretik, ez ostean), beraz hirugarren elementua ezabatzeko garaian eraginkortasunean irabaz daiteke.

OHARRA: Ezinbestekoa da soluzioa eraginkorra izatea. Zehazki, elementuen ordenak garrantzirik ez izateaz probetxua aterako dugu begiztak ekiditeko.

6. Ariketa: Ezabatu hirugarren elementua bektore ordenatuan

Ariketa hau **ADAz** bakarrik ebatzi behar da.
Fitxategia (ADA): bektoreak.ads eta idatzi_zerrenda.adb (ez dira moldatu behar).
Txantiloiak (ADA): ezabatu_hirugarren_elementua_ordenatuta.adb eta proba_ezabatu_hirugarren_elementua_ordenatuta.adb

Gehienez N elementu izango dituen osoko-bektore bat emanda (N > 0 eta gorako ordenan ordenatuta), hirugarren elementua ezabatzen duen azpiprograma inplementatu. Aurreko ariketaren antzekoa da, baina kasu honetan bektorearen elementuak ordenatuta egon behar dute, eta ezabaketaren ostean ere ordenatuta jarraitu behar dute. Bektorean hiru elementu baina gutxiago badaude, orduan ez da eze egingo.

7. Ariketa: Txertatu zenbaki bat zerrendako posizio batean

Ariketa hau **ADAz** bakarrik ebatzi behar da.
Fitxategia (ADA): bektoreak.ads eta idatzi_zerrenda.adb (ez dira moldatu behar).
Txantiloiak (ADA): txertatu_posizioan.adb eta proba_txertatu_posizioan.adb

Gehienez N-1 elementu dituen bektore bat izanda (ondorioz, gutxienez elementu batentzat tokia dauka), azpiprograma bat inplementatu zeinak *Zen* elementua *Pos* posizioan txertatuko duen, beharrezko elementuak eskuinera mugituz. Gogoratu aurretik eskuinera_mugitu azpiprograma garatu duzuela. Aurreko ariketetan bezala, Osokoen_Zerrenda erabili beharko duzue Osokoen_Bektorea beharrean.

8. Ariketa: Posizio baten bilaketa matrize batean

Ariketa hau **Pythonez** bakarrik ebatzi behar da.
Txantiloiak (Python): posizioa_matrizean.py

Zenbaki osoko bat eta N x M (N>0 eta M>0) osokoen matrize bat emanda, zenbakia matrize horretan duen posizioa itzultzen digun azpiprograma inplementatu (hau da, X,Y koordenatuak). Zenbakia matrizean ez balego zenbakiaren posizioa -1,-1 izango da. Zenbakia errepikatuta balego, lehenengo posizioa itzuli behar da.

9. Ariketa: Matrize baten maximoa

Ariketa hau **ADAz** bakarrik ebatzi behar da.
Fitxategia (ADA): matrizeak.ads (ez da moldatu behar).
Txantiloiak (ADA): maximoa_matrizean.adb eta proba_maximoa_matrizean.adb

N x M (N>0 eta M>0)osokoen matrize bat emanda, matrize horretan dagoen elementurik handiena zein den eta bere posizioa kalkulatzen duen azpiprograma inplementatu. Zenbakia errepikaturik balego, lehenengo posizioa itzuli.

10. Ariketa: Txertaketa bidezko ordenazioa

Ariketa hau **Pythonez** bakarrik ebatzi behar da.
Txantiloiak (Python): txertaketa_bidezko_ordenazioa.py

N osoko dituen bektorea emanda (N>0), ordenatzeko txertaketa bidezko metodoa jarraitzen duen azpiprograma inplementatu. Soluzioak, N > 0 elementutako edozein bektorerentzat balio behar du.

- Laguntza bideoa: http://www.youtube.com/watch?v=gTxFxgvZmQs&feature=related

Bektorea zeharkatzen hasiko gara elementuz elementu. *i*-garren iterazioaren hasieran, *i*-1 elementuak ordenatuta egongo dira; *i+1*-garren iterazioan uneko elementua hartuko da eta aurreko elementuekin elementuz elementu konparatuko da (alegia, jada ordenatuta dauden elementuekin) topatzeko zein litzateke egungo elementuaren posizio egokia. Ondorioz, kasu gehienetan bektorean hutsune bat sortu beharko dugu elementu batzuk eskuinera mugituz, eta uneko elementua txertatu ahal izateko.

Ariketa hau egiteko honako azpiprogramak inplementatzea eskatzen da:

- *bilatu_txertatze_posizioa*. Parametro moduan B bektorea, uneko posizioa (ordenatuta dauden elementuak zehazten du) eta uneko elementua (txertatu nahi den zenbakia); eta uneko elementuari dagokion posizioa itzultzen du (0 eta uneko posizioa-1 tartean),
- *mugitu_posizio_bat_eskuinera*. Parametro moduan B bektorea eta 2 zenbaki osoko jasotzen ditu bi posizio adieraziz: uneko posizioa (lehen bezala, ordenatuta dauden elementuak zehazten ditu), eta hasierako posizioa (gero elementua berria txertatu nahiko dugun posizioa adierazten du). Bektorea bera itzultzen du, hasierako posizioa eta uneko posizioaren artean dauden elementu guztiak eskuinera mugituta dituelarik. Kontuan izan, uneko posizioa berridatziko dugula hutsunea sortzeko, eta ondorioz funtzio honi deitu aurretik uneko elementua gordetzea komeni da.

Adibidea:

Hasierako zerrenda (10 elementu):

0	1	2	3	4	5	6	7	8	9
9	5	3	4	10	8	13	24	15	11

Txertaketa 1: lehenengo elementua (9) 0. posizioan txertatzen da; bektorea ez da aldatzen (unekoPos= 1):

0	1	2	3	4	5	6	7	8	9
9	5	3	4	10	8	13	24	15	11

Txertaketa 2: bigarren elementua (5) 0. posizioan txertatzen da, 9 zenbakia eskuinera mugituz (unekoPos=2):

0	**1**	2	3	4	5	6	7	8	9
5	**9**	3	4	10	8	13	24	15	11

Txertaketa 3: hirugarren elementua (3) 0. posizioan txertatzen da, 5 eta 9 zenbakiak eskuinera mugituz (unekoPos=3):

0	**1**	**2**	3	4	5	6	7	8	9
3	**5**	**9**	4	10	8	13	24	15	11

Txertaketa 4: laugarren elementua (4) 1. posizioan txertatzen da, 5 eta 9 zenbakiak eskuinera mugituz (unekoPos=4):

0	**1**	**2**	**3**	4	5	6	7	8	9
3	**4**	**5**	**9**	10	8	13	24	15	11

Txertaketa 5: bosgarren elementua (10) 4. posizioan txertatzen da, ezer mugitu gabe (unekoPos=5):

0	**1**	**2**	**3**	**4**	5	6	7	8	9
3	**4**	**5**	**9**	**10**	8	13	24	15	11

etab...

11. Ariketa: Hautaketa bidezko ordenazioa

Ariketa hau **Pythonez** bakarrik ebatzi behar da.
Txantiloiak (Python): hautaketa_bidezko_ordenazioa.py

N osoko dituen bektorea emanda (N>0), ordenatzeko hautaketa bidezko metodoa jarraitzen duen azpiprograma inplementatu. Soluzioak, N > 0 elementutako edozein bektorerentzat balio behar du.

- Laguntza bideoa: http://www.youtube.com/watch?v=boOwArDShLU

Algoritmo honek honela funtzionatzen du: elementu txikiena bilatzen da, eta lehenengo posizioarenarekin ordezkatzen da. Ostean, gainerako bektorean uneko minimoa bilatzen da, eta bigarren posizioarenarekin ordezkatzen da. Orokorrean, i. iterazioan, lehenengo i-1 elementuak ordenatuta daude eta euren posizio definitiboan daude dagoeneko; elementu txikiena bilatzen da i. eta bektorearen amaieraren artean, eta uneko posizioarengatik ordezkatzen da.

Ariketa hau egiteko honako azpiprogramak inplementatzea eskatzen da:

- *bilatu_minimoaren_posizioa*. Parametro moduan B bektorea eta hasierako posizioa (uneko posizioarekin bat etorriko da, jadanik ordenatuta dauden elementuak zehazteko balio du). Balioa minimoa dagoen elementuaren posizioa itzultzen du, hasierako posizioaren eta bektorearen bukaeraren artean dagoena.
- *trukatu*. Parametro moduan B bektorea eta 2 zenbaki osoko jasotzen ditu bi posizio adieraziz: *posA* eta *posB*. Bektorea bera itzultzen du, *posA* eta *posB* posiziotan dauden elementuak elkartrukatuta dituelarik.

Adibidea:

Hasierako zerrenda (10 elementu):

0	1	**2**	3	4	5	6	7	8	9
9	5	**3**	4	10	8	13	24	15	11

0 eta azken posizioen arteko elementu minimoa 3 da (2 posizioa); 0 eta 2 posizioak trukatzen dira:

0	1	2	**3**	4	5	6	7	8	9
3	5	9	**4**	10	8	13	24	15	11

1 eta azken posizioen arteko elementu minimoa 4 da (3 posizioa); 1 eta 3 posizioak trukatzen dira:

0	**1**	2	**3**	4	5	6	7	8	9
3	**4**	9	**5**	10	8	13	24	15	11

2 eta azken posizioen arteko elementu minimoa 5 da (3 posizioa); 2 eta 3 posizioak trukatzen dira:

0	**1**	**2**	3	4	**5**	6	7	8	9
3	**4**	**5**	9	10	**8**	13	24	15	11

3 eta azken posizioen arteko elementu minimoa 8 da (5 posizioa); 3 eta 5 posizioak trukatzen dira:

0	**1**	**2**	**3**	4	**5**	6	7	8	9
3	**4**	**5**	**8**	10	**9**	13	24	15	11

4 eta azken posizioen arteko elementu minimoa 9 da (5 posizioa); 4 eta 5 posizioak trukatzen dira:

0	**1**	**2**	**3**	**4**	5	6	7	8	9
3	**4**	**5**	**8**	**9**	10	13	24	15	11

Etab...

12. Ariketa: Burbuila bidezko ordenazioa

Ariketa hau **Pythonez** bakarrik ebatzi behar da.
Txantiloiak (Python): burbuila_bidezko_ordenazioa.py

N osoko dituen bektorea emanda (N>0), ordenatzeko burbuila bidezko metodoa modu eraginkorrean jarraitzen duen azpiprograma inplementatu. Soluzioak, N > 0 elementutako edozein bektorerentzat balio behar du.

- Laguntza bideoa: http://www.youtube.com/watch?v=1JvYAXT_064&feature=related

Oinarrizko algoritmoa honakoa izan daiteke:

```
ADAz honela izango litzateke
zen_pasatakoak:=1;
loop exit when zen_pasatakoak > zen_elementuak-1;
    i:=1;
    loop exit when i> zen_elementuak-1;
          if t(i) > t(i+1) then ---aldatu
                lag:=t(i);
                t(i):=t(i+1);
                t(i+1):=lag;
          end if;
          i:=i+1;
    end loop;
zen_pasatakoak:= zen_pasatakoak+1;
end loop;
```

Adibidea:

Hasierako zerrenda (4 elementu):

7	5	1	2

Lehenengo iterazioaren ostean, azkene elementua ordenatuta dago.

5	1	2	7

Bi iterazio ostean, bi elementu ordenatuta daude.

1	2	5	7

Eta hirugarren eta azken iterazioaren ostean bektorea berdin gelditukо litzateke.

1	2	5	7

Azken iterazioan ez da ezer egin, zerrenda dagoeneko ordenatuta dagoelako. Hala ere, kasurik okerrenean (zerrenda ERABAT desordenatuta egon izan balitz...), azken iterazioa ere beharrezkoa izango zen. Aurretik ezin daitekeenez jakin iterazio guztiak beharrezkoak izango diren, ala ez, algoritmoak urrats guztiak ematen ditu.

Ariketa honetan, zuen soluzioak aurreko algoritmoaren **eraginkortasuna hobetu beharko du.** Horretarako, boolear bat erabili dezakezu zeinak iterazio batean aldaketarik egin den ala ez jasoko duen. Uneko iterazioan ez badu aldaketarik egin, zerrenda erabat ordenatuta dagoelako izango da.