Programarea Algoritmilor
– LABORATOR NR. 4 –

Funcții, generatori. Module. Tratarea excepțiilor

1. (Modulul `math`, funcția `math.sqrt`)
a) Scrieți o funcție "ipotenuza" care primește ca parametri două numere *a* și *b* și furnizează ipotenuza triunghiului dreptunghic având catetele de lungimi *a* și *b*.
b) Se citește de la tastatură un număr natural *b*. Folosind funcția de mai sus, scrieți pe fiecare linie din fișierul text "triplete_pitagoreice.txt" câte 3 numere a, b și c, astfel: valoarea lui b este cea citită de la tastatură, a este un număr natural mai mic sau egal decât b, iar *c* este tot un *număr natural* reprezentând ipotenuza triunghiului dreptunghic având catetele *a* și *b*. ***Indicație:*** Pentru un anumit b, trebuie găsite *toate* valorile posibile pentru a și c care respectă condițiile de mai sus.
Exemple de triplete pitagoreice (pentru valori diferite ale lui b): (3,4,5), (5,12,13), (8,15,17) etc.

2. (Modulul `math`, constanta `math.pi`)
a) Scrieți o funcție "lungime_arie_cerc" care primește ca parametru un număr real *r* și returnează lungimea și aria cercului având raza de lungime *r*.
b) Se citește de la tastatură un număr natural *r* reprezentând lungimea razei unui cerc. Folosind funcția de mai sus, să se afișeze pe ecran lungimea și aria cercului de rază *r*.

3. (Tratarea excepțiilor)
a) Să se scrie o funcție "min_max" care primește un număr *variabil* de parametri (numere naturale) și returnează cel mai mic și cel mai mare număr dintre cele primite ca parametri, dacă există cel puțin un parametru și dacă toți parametrii sunt numere naturale, sau returnează `None` altfel.
b) Să se citească tot conținutul fișierului text "numere.txt" și apoi să se afișeze pe ecran rezultatele obținute aplicând funcția "min_max" asupra sa. Dacă valoarea returnată de funcția min_max este diferită de `None`, se va scrie în fișierul text "impartire.txt" rezultatul împărțirii valorii maxime din fișierul text la cea minimă. Să se trateze excepțiile care pot să apară: nu există fișierul text de intrare, nu există drept de scriere pentru fișierul de ieșire, fișierul de intrare conține valori care nu sunt numere naturale, împărțire la zero etc.
Exemplu: Dacă fișierul text "numere.txt" conține: 11 9 31 7 145 5 101 4 80, atunci funcția "min_max" va returna (4, 145), iar în fișierul "impartire.txt" se va scrie: 36.25

4. (Generator)
Implementați un generator infinit de numere prime. Folosind acest generator, scrieți un program care citește de la tastatură un număr natural nenul n≥2 și afișează pe ecran:
a) numerele prime cel mult egale cu n;
b) primele n numere prime.

5. Scrieți o funcție "negative_pozitive" care primește ca parametru o lista de numere întregi și returnează două liste, prima formată din numerele strict negative din lista primită ca parametru și a doua formată din cele strict pozitive. Scrieți un program care citește de la tastatură numele unui fișier text și apoi, folosind apeluri utile ale funcției "negative_pozitive" scrie la sfârșitul fișierului text, pe două linii separate, numerele strict negative, respectiv cele strict pozitive, ordonate crescător.

6. Fie *v* o listă formată din n numere întregi. ***În cadrul unui modul*****,** definiți complet următoarele funcții**:**

a) citire – citește valoarea lui n și apoi cele n elemente ale listei v;

b) afișare – afișează elementele listei v pe o linie, despărțite prin câte un spațiu;

c) valpoz – construiește o listă formată din valorile pozitive din v;

d) semn – schimbă semnul fiecărui element al tabloului v.

Scrieți un program care citește lista v și afișează pe o linie valorile negative din lista v și pe o altă linie valorile pozitive din lista v, folosind apeluri utile ale funcțiilor definite anterior.

7. (Sortare cu parametrii `key`=fct_comparator și/sau `reverse`=True/False)

Din fișierul "cuvinte.txt" se citesc cuvinte care se salvează într-o listă L. În fișierul "cuv_sortate.txt" să se scrie, pe câte o linie, lista L sortată astfel:

a) descrescător, folosind compararea implicită

b) crescător, comparând după lungimea cuvintelor și apoi după ordinea alfabetică

c) crescător după lungimea cuvintelor, dar pentru cuvintele de aceeași lungime păstrând ordinea din lista originală.

Exemplu:

Dacă L = ["zi", "ana", "sac", "acadea", "bac", "nori", "zar", "mi", "abur"], fișierul cuv_sortate.txt va avea următorul conținut:

a) ['zi', 'zar', 'sac', 'nori', 'mi', 'bac', 'ana', 'acadea', 'abur']

b) ['mi', 'zi', 'ana', 'bac', 'sac', 'zar', 'abur', 'nori', 'acadea']

c) ['zi', 'mi', 'ana', 'sac', 'bac', 'zar', 'nori', 'abur', 'acadea']

8. Fișierul text "angajati.txt" conține pe fiecare linie numele, vârsta și salariului câte unui angajat, despărțite prin virgule. Scrieți un program care să încarce datele din fișierul text într-o listă și să afișeze următoarele informații:

a) informațiile despre un angajat al cărui nume se citește de la tastatură (pe ecran);

b) salariul maxim și numele angajaților care au salariul respectiv (pe ecran);

c) salariul mediu din firmă (pe ecran);

d) angajații sortați alfabetic după nume (în fișierul text "angajati_nume.txt");

e) angajații sortați descrescător după vârstă și apoi alfabetic după nume (în fișierul text "angajati_varsta_nume.txt");

f) angajații sortați descrescător după salariu, apoi crescător după vârstă (în fișierul text "angajati_salariu_varsta.txt").

Fiecare cerință se va rezolva în cadrul unei funcții!

9. a) Scrieți o funcție care să citească de la tastatură o listă cu elemente numere întregi. Numărul de elemente ale listei și elementele sale se vor citi în cadrul funcției.

b) Scrieți o funcție care să returneze poziția primului element mai mare decât un număr întreg x dintr-o secvență a unei liste cuprinsă între doi indici i și j ($0 \le i \le j < n$) sau valoarea -1 dacă nu există niciun astfel de element.

c) Scrieți un program care, folosind apeluri utile ale funcțiilor definite anterior, afișează mesajul "Da" în cazul în care o listă de numere întregi, citită de la tastatură, este sortată strict descrescător sau mesajul "Nu" în caz contrar.

10. a) Entitatea Student este caracterizată prin următoarele informații: nume, grupă, numărul total de credite obținute și situația școlară a unui student (Promovat/Nepromovat). Scrieți o funcție care să determine situațiile școlare a *n* studenți ale căror date (nume, grupă și numărul total de

credite) sunt memorate într-o listă t cu elemente de tip Student. Un student este considerat promovat dacă a obținut un număr de credite cel puțin egal cu un număr natural nenul c.
b) Scrieți o funcție care să sorteze elementele unei liste t, formată din *n* elemente de tip Student în ordinea crescătoare a grupelor, iar în cadrul fiecărei grupe studenții se vor ordona descrescător în ordinea numărului total de credite obținute. (Folosiți eventual un dicționar având ca chei numerele grupelor, iar ca valori liste cu colecții care conțin restul informațiilor despre studenții din acea grupă).

11. a) Scrieți o funcție cu număr variabil de parametri care să furnizeze numărul natural obținut prin alipirea cifrelor maxime ale numerelor naturale nenule primite ca parametri.
De exemplu, pentru numerele 4251, 73, 8 și 133 funcția trebuie să returneze numărul 5783.
b) Scrieți o funcție cu 3 parametri nenuli de tip întreg a,b și c care să verifice dacă aceștia pot fi considerați ca fiind numere scrise în baza 2 sau nu, folosind apeluri utile ale funcției definite anterior. ***De exemplu***, pentru numerele 1001, 11 și 100 funcția trebuie să returneze valoarea True, iar pentru numerele 1001, 17 și 100 trebuie să returneze valoarea False.

12. a) Scrieți o funcție generică de căutare având următorul antet: cautare(x, L, cmpValori)
Funcția trebuie să returneze indexul ultimei apariții a valorii x în lista L sau None dacă valoarea x nu se găsește în listă. Funcția comparator cmpValori se consideră că returnează True dacă valorile primite ca parametri sunt egale sau False în caz contrar.
b) Scrieți o funcție care să afișeze, folosind apeluri utile ale funcției *cautare*, mesajul DA în cazul în care o listă L formată din n numere întregi este palindrom sau mesajul NU în caz contrar. O listă este *palindrom* dacă prin parcurgerea sa de la dreapta la stânga se obține aceeași listă.
De exemplu, lista L=[101,17,101,13,5,13,101,17,101] este palindrom.

13. Scrieți o funcție cu număr variabil de parametri care să caute un cuvânt dat în mai multe fișiere text. Funcția va scrie într-un fișier text câte o linie pentru fiecare fișier text de intrare, astfel: numele fișierului text de intrare și apoi numerele de ordine ale liniilor pe care apare cuvântul dat în acel fișier sau un mesaj corespunzător dacă fișierul nu conține cuvântul respectiv.

Indicații de rezolvare:
a) Antetul funcției va fi: cautare_cuvant(cuv, nume_fis_out, *nume_fis_in)
b) Parametrul fix *cuv* reprezintă cuvântul pe care îl vom căuta, parametrul fix *nume_fis_out* reprezintă numele fișierului text în care se vor scrie rezultatele căutării, iar lista parametrilor variabili va conține numele fișierelor text în care va fi căutat cuvântul respectiv.

De exemplu, prin apelul
cautare_cuvant("lac","rez.txt", "eminescu.txt", "bacovia.txt")
se va căuta cuvântul “lac” în fișierele text “eminescu.txt” și “bacovia.txt”,
iar rezultatul căutării va fi scris în fișierul text “rez.txt”.

c) În cadrul funcției, preluăm, pe rând, numele fiecărui fișier text din lista parametrilor variabili și efectuăm următoarele operații:

- deschidem fișierul text curent;
- parcurgem fișierul linie cu linie și căutăm cuvântul dat pe linia curentă;
- dacă linia curentă conține cuvântul căutat, atunci scriem numărul de ordine al liniei curente în fișierul de ieșire, marcăm faptul că am găsit cuvântul în fișierul curent folosind o variabilă booleană și întrerupem căutarea pe linia curentă;
- după ce am terminat de parcurs fișierul curent, verificăm dacă am găsit sau nu cuvântul căutat în el și, în caz negativ, scriem un mesaj corespunzător;
- închidem fișierul curent.

14. Traseu

Ana locuiește în București și își dorește foarte mult să ajungă la prietena sa Maria care locuiește în Brașov. Într-un fișier text, Ana are programul mai multor curse de autocare, sub forma indicată în exemplul de mai jos.

Realizați următoarele funcții:

a) `fisier_dict(nume_fisier)` - primește ca parametru numele unui fișier text conținând programul unor curse de autocare și întoarce o listă de dicționare având următoarea formă: [{'plecare':'Bucuresti', 'sosire' : 'Brasov', 'ora_plecare' : '9:20', 'ora_sosire': '13:50'}, {'plecare' : 'Sinaia', 'sosire' : 'Brasov', 'ora_plecare' : '11:20', 'ora_sosire' : '15:00'}];

b) `timp_calatorie(ora_plecare, ora_sosire)` - primește ca parametrii ora de plecare și ora de sosire a unei curse și întoarce timpul necesar pentru călătorie. Dacă acesta va depăși 23 de ore și 59 de minute funcția va afișa un mesaj de eroare și va întoarce valoarea `None`;

c) `calatorie(lista_program)` - primește ca parametru lista returnată de funcția `fisier_dict` și returnează o nouă listă de dicționare conținând cursele care o pot ajuta pe Ana să ajungă la destinație, direct sau indirect.

d) `scrie_fisier(lista_dict)` - scrie în fișierul "traseu.txt" toate cursele de la București la Brașov;

e) `timp_minim(lista_dict)` - returnează traseul cu timp minim.

Folosind apeluri utile ale funcțiilor de mai sus, realizați un algoritm care să o ajute pe Ana să ajungă la prietena sa Maria folosind un traseu care să implice maxim o schimbare.

Exemplu:

program.txt

Bucuresti -> Brasov 10:00 17:00
Bucuresti -> Ploiesti 12:15 14:00
Ploiesti -> Comarnic 12:45 16:00
Ploiesti -> Brasov 15:00 19:30
Targoviste -> Brasov 16:00 19:00
Bucuresti -> Sinaia 16:15 19:20
Sinaia -> Brasov 20:00 22:15

traseu.txt

Plecare la ora 10:00 sosire ora 17:00 durata 7h
Plecare la ora 12:15 schimbare in Ploiesti sosire 19:30 7h 15min
Plecare la ora 16:15 schimbare in Sinaia sosire 22:15 6h

Timp minim
Bucuresti Sinaia Brasov

15. Ping

Într-un fișier text se află mesajul returnat de o comandă de tip ping. Realizați următoarele funcții:

a) `determina_ip(nume fisier)` - primește ca parametru numele unui fișier text de tipul indicat mai sus și returnează adresa IP și DNS (numele domeniului) către care s-a dat comanda ping;

b) `nr_pachete(nume_fisier)` - primește ca parametru numele unui fișier text de tipul indicat și returnează numărul de pachete trimise, respectiv numărul de linii de forma
64 bytes from 216.58.214.238: icmp_seq=0 ttl=56 time=16.453 ms

c) `verificare_timp(nume_fisier)` - primește ca parametru numele unui fișier text și returnează mesaje de eroare daca timpii prezenți pe ultima linie nu sunt egali cu timpii din fișier.

Folosind apeluri utile ale funcțiilor de mai sus, realizați un algoritm ce scrie în fișierul "ping.out" numărul de pachete trimise, timpul minim, maxim, mediu, IP și DNS.
Rezolvați problema folosind expresii regulate (modulul `re`)***.***

Exemplu:
ping.in

```
192-168-0-2:~ mihaela-dianabaldovin$ ping google.com -c 5
PING google.com (216.58.214.238): 56 data bytes
64 bytes from 216.58.214.238: icmp_seq=0 ttl=56 time=16.453 ms
64 bytes from 216.58.214.238: icmp_seq=1 ttl=56 time=16.136 ms
64 bytes from 216.58.214.238: icmp_seq=2 ttl=56 time=15.838 ms
64 bytes from 216.58.214.238: icmp_seq=3 ttl=56 time=16.669 ms
64 bytes from 216.58.214.238: icmp_seq=4 ttl=56 time=15.775 ms
--- google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 15.775/16.174/16.669/0.345 ms
```

ping.out

```
Timp minim: 15.775 ms
Timp maxim: 16.669 ms
Timp mediu: 16.174 ms
IP: 216.58.214.238
DNS: google.com
```