Laborator 4

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. (Generator)

Implementați un generator infinit de numere prime. Folosind acest generator, scrieți un program care citește de la tastatură un număr natural nenul $n \geq 2$ și afișează pe ecran:

a) numerele prime cel mult egale cu n;

b) primele n numere prime.

3. 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.

4. 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.

5. 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 ($0 \leq i \leq j < n$) 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.

6.

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.

7. 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.

8. (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']

9.

a) Să se implementeze o funcție care primește ca și parametru o valoare întreagă k și returnează o funcție F care primește ca parametru un număr întreg x. F(x) va returna true dacă x < k, False altfel.

b) Folosind funcțiile de la punctul a, să se implementeze un algoritm de sortare.