Implementaţi o aplicaţie în limbajul C ce rezolvă probleme de gestionare unei reţele de depozite pentru stocarea de materiale de constructii sau bunuri de larg consum.

1. Scrieţi secvenţa de cod sursă pentru crearea unei structuri de tip ***Listă dublu inlantuita*** ce conţine date aferente unor depozite. În fiecare nod al listei, depozitele se stochează la nivel de pointer/adresă (elemente de tip ***Depozit****). Inserarea unui nod are loc astfel incât lista dublu inlantuita să fie ordonată crescător in funcţie de id depozit (inserare nod în interiorul listei). Inserarea unui nod se implementează într-o funcţie care se apelează în secvenţa de creare a structurii ***Listă dublu inlantuita***.

   Structura ***Depozit*** este definită astfel:

```
struct Depozit {
      int id;
      char* denumire;
      char* localitate;
      float suprafata;
      float capacitate_stocare;
};
```

   Exemplu set de date pentru un depozit: **{11, „Amazon B2”, „Bucuresti”, 327.59, 2000.24}**

   Lista simplă va conţine datele a cel puţin 7 depozite care se preiau ca input dintr-un fişier text. **(2p)**

2. Scrieţi şi apelaţi funcţia pentru modificarea denumirii unui depozit specificat prin id. **(1p)**

3. Scrieţi şi apelaţi funcţia determină numărul mediu de angajaţi per depozit dintr-o localitate specificată. **(1p)**

4. Scrieţi secvenţa de cod care copiază datele din ***Lista dublu inlantuite*** creată anterior într-o structură ***arbore binar de cautare***. Cheia de inserare in arbore este ***id din structura Depozit***. **(2p)**

5. Scrieţi şi apelaţi funcţia pentru modificarea localităţii (cheie de căutare - id) unui depozit în arborele binar. Depozitul este specificat prin id (cheie de căutare). **(2p)**

6. Scrieţi secvenţa de cod care dezalocă structurile ***Listă dublu inlantuita*** şi ***Arborele binar de cautare*** create la punctele anterioare. **(1p + 1p)**

OBSERVAŢIE: Implementările plagiate vor fi evaluate cu 0 puncte, indiferent de sursă.