Problema 1:

La un turneu de tenis participa n jucatori cu numerele de ordine de la 1 la n. Fiecare din ei joaca cu toto ceilalti cate o partida. Fiecare dintre partide este data ca o pereche de jucatori (i, j) cu semnificatia ca jucatorul i l-a invins pe jucatorul j. Afisati primii 3 jucatori in ordine descrescatoare a numarului de victorii din turneu.

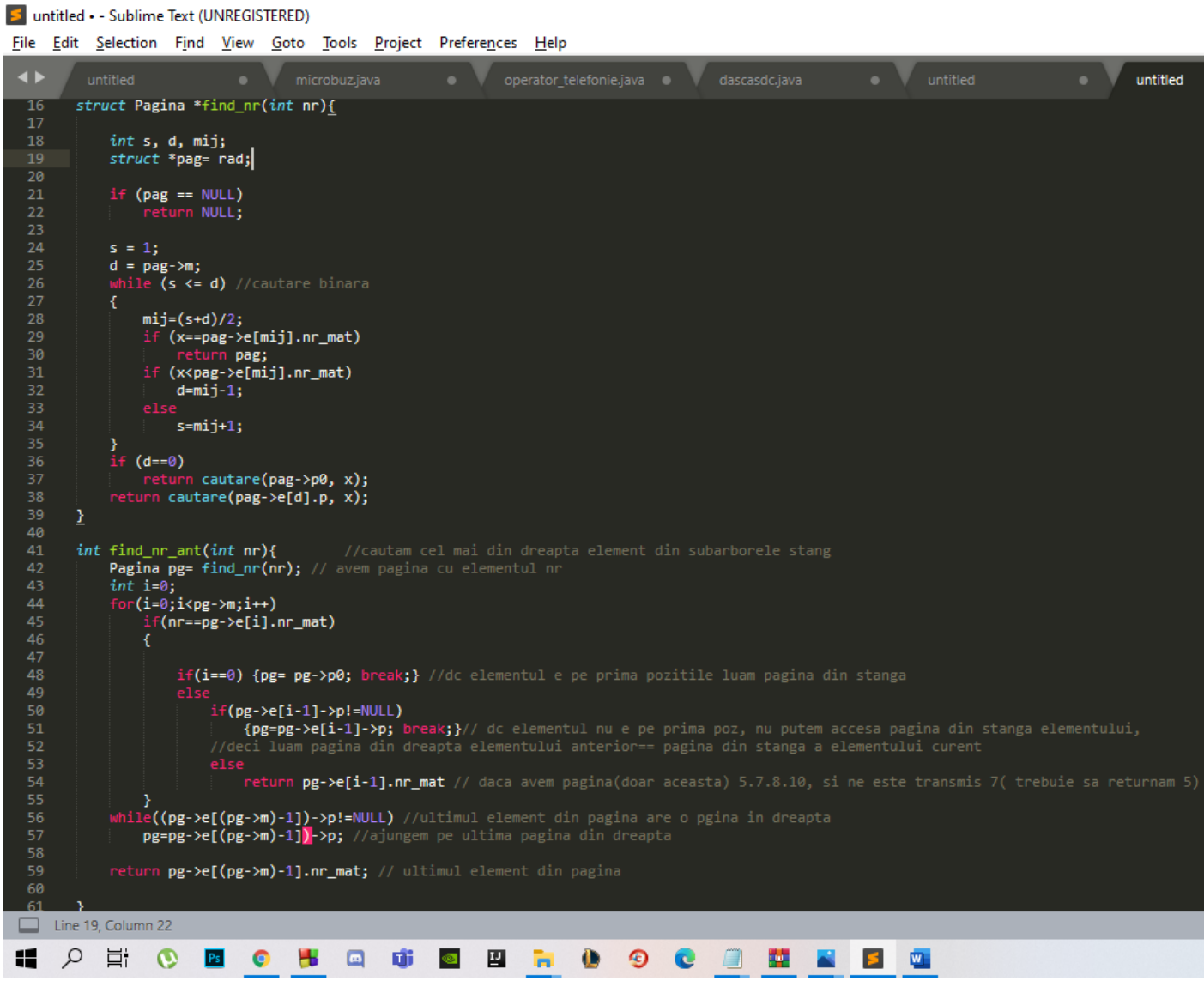
Exemplu:

Intrari
5
1 2 (cu semnificatia ca jucatorul 1 l-a invins pe jucatorul 2)
3 2
3 4
1 4
5 1
3 5
5 4
3 1
5 2
2 4
Iesiri
3 4 (cu semnificatia ca jucatorul 3 are 4 victorii)
5 3
1 2

Problema 2:

Pentru o serie de studenti, se cunosc numarul matricol si anul nasterii. Considerand ca pentru stocarea datelor despre studenti se foloseste o structura de tip Arbore B, sa se scrie o functie care primind un numar matricol returneaza numarul matricol imediat mai mic valoric din serie.

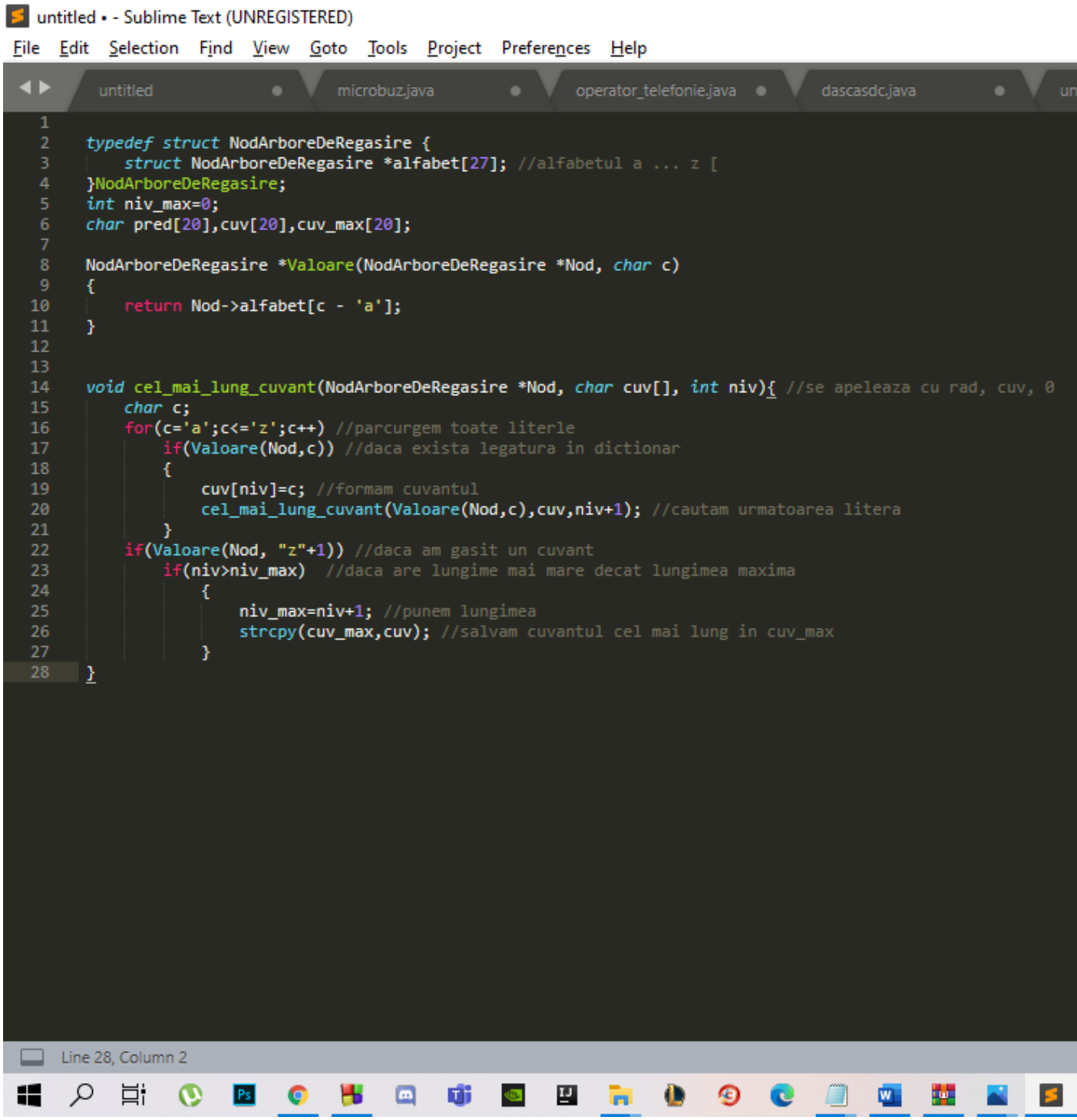
```
struct Nod
{
        int nr_mat;
        struct Pagina* p;  //pagina cu chei mai mari decat cheia curenta
        int contor;
};


struct Pagina
{
        int m;  //cate de elemente a paginii
        struct Pagina* p0; //pagina cu chei mai mici decat in pagina curenta
        Nod e[nn+1]; //
}*rad;
```




```
16  struct Pagina *find_nr(int nr){
17
18      int s, d, mij;
19      struct *pag= rad;
20
21      if (pag == NULL)
22          return NULL;
23
24      s = 1;
25      d = pag->m;
26      while (s <= d) //cautare binara
27      {
28          mij=(s+d)/2;
29          if (x==pag->e[mij].nr_mat)
30              return pag;
31          if (x<pag->e[mij].nr_mat)
32              d=mij-1;
33          else
34              s=mij+1;
35      }
36      if (d==0)
37          return cautare(pag->p0, x);
38      return cautare(pag->e[d].p, x);
39  }
40
41  int find_nr_ant(int nr){        //cautam cel mai din dreapta element din subarborele stang
42      Pagina pg= find_nr(nr); // avem pagina cu elementul nr
43      int i=0;
44      for(i=0;i<pg->m;i++)
45          if(nr==pg->e[i].nr_mat)
46          {
47
48              if(i==0) {pg= pg->p0; break;} //dc elementul e pe prima pozitile luam pagina din stanga
49              else
50                  if(pg->e[i-1]->p!=NULL)
51                      {pg=pg->e[i-1]->p; break;}// dc elementul nu e pe prima poz, nu putem accesa pagina din stanga elementului,
52                  //deci luam pagina din dreapta elementului anterior== pagina din stanga a elementului curent
53                  else
54                      return pg->e[i-1].nr_mat // daca avem pagina(doar aceasta) 5.7.8.10, si ne este transmis 7( trebuie sa returnam 5)
55          }
56      while((pg->e[(pg->m)-1])->p!=NULL) //ultimul element din pagina are o pgina in dreapta
57          pg=pg->e[(pg->m)-1])->p; //ajungem pe ultima pagina din dreapta
58
59      return pg->e[(pg->m)-1].nr_mat; // ultimul element din pagina
60
61  }
```

Problema 3:

Fie un dictionar implementat cu ajutorul unui arbore de regasire. Sa se scrie o functie care determina cuvantul cel mai lung.



Problema 4:

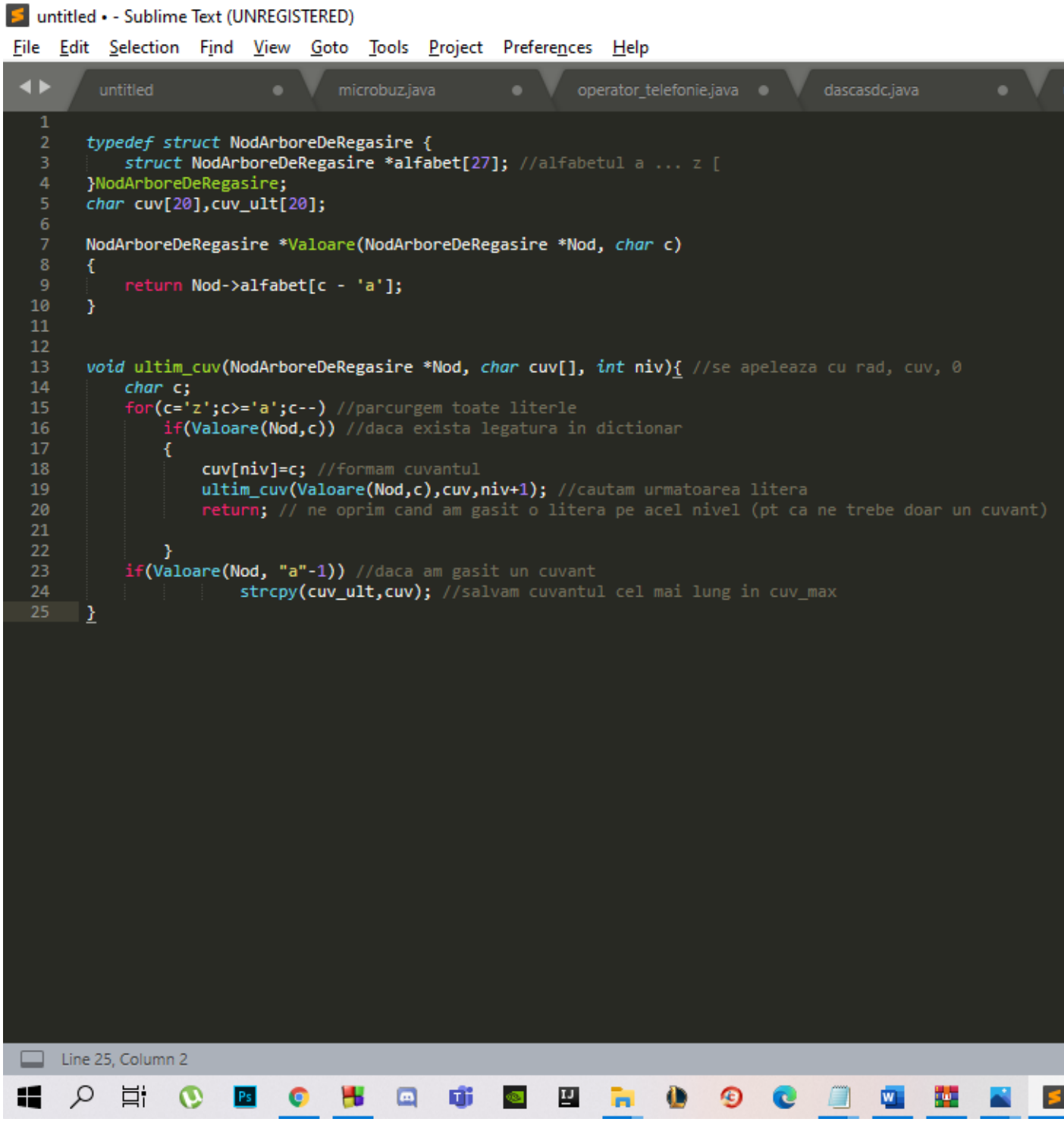
Fie T un tablou cu N pacienti ai unui spital, pentru care se cunosc ID-ul (intreg) si anul nasterii. Sa se stocheze pacientii dati intr-un arbore binar de inaltime minima.

Problema 5:

Pentru pacientii unui spital se cunosc ID-ul (intreg) si anul nasterii. Sa se scrie o functie pentru stergerea tuturor pacientilor nascuti dupa anul 2010 folosind un Arbore B.

Problema 6:

Fie un dictionar implementat cu ajutorul unui arbore de regasire. Sa se scrie o functie care efectueaza afisarea ultimului cuvant din dictionar.




```c
typedef struct NodArboreDeRegasire {
    struct NodArboreDeRegasire *alfabet[27]; //alfabetul a ... z [
}NodArboreDeRegasire;
char cuv[20],cuv_ult[20];

NodArboreDeRegasire *Valoare(NodArboreDeRegasire *Nod, char c)
{
    return Nod->alfabet[c - 'a'];
}


void ultim_cuv(NodArboreDeRegasire *Nod, char cuv[], int niv){ //se apeleaza cu rad, cuv, 0
    char c;
    for(c='z';c>='a';c--) //parcurgem toate literle
        if(Valoare(Nod,c)) //daca exista legatura in dictionar
        {
            cuv[niv]=c; //formam cuvantul
            ultim_cuv(Valoare(Nod,c),cuv,niv+1); //cautam urmatoarea litera
            return; // ne oprim cand am gasit o litera pe acel nivel (pt ca ne trebe doar un cuvant)

        }
    if(Valoare(Nod, "a"-1)) //daca am gasit un cuvant
                strcpy(cuv_ult,cuv); //salvam cuvantul cel mai lung in cuv_max
}
```

Problema 7:

Pentru angajatii unei firme se cunosc ID-ul (intreg) si anul angajarii. Stiind ca un angajat poate avea mai multi subalterni directi, dar un singur sef direct, angajatii se stocheaza intr-un arbore generalizat dat prin tablouri de primul fiu si frate drept, in care indicatori respectivi retin indexul corect din tabloul de angajati. Sa se scrie o functie pentru afisarea tuturor angajatilor care nu au subalterni.

Problema 8:

Pentru o serie de studenti se cunosc: numele, prenumele, nota la PAA. Considerand ca pentru stocarea datelor despre studenti se foloseste o structura de tip arbore binar ordonat, sa se scrie o functie pentru afisarea tuturor studentilor a caror nota este in intervalul [7, 8).

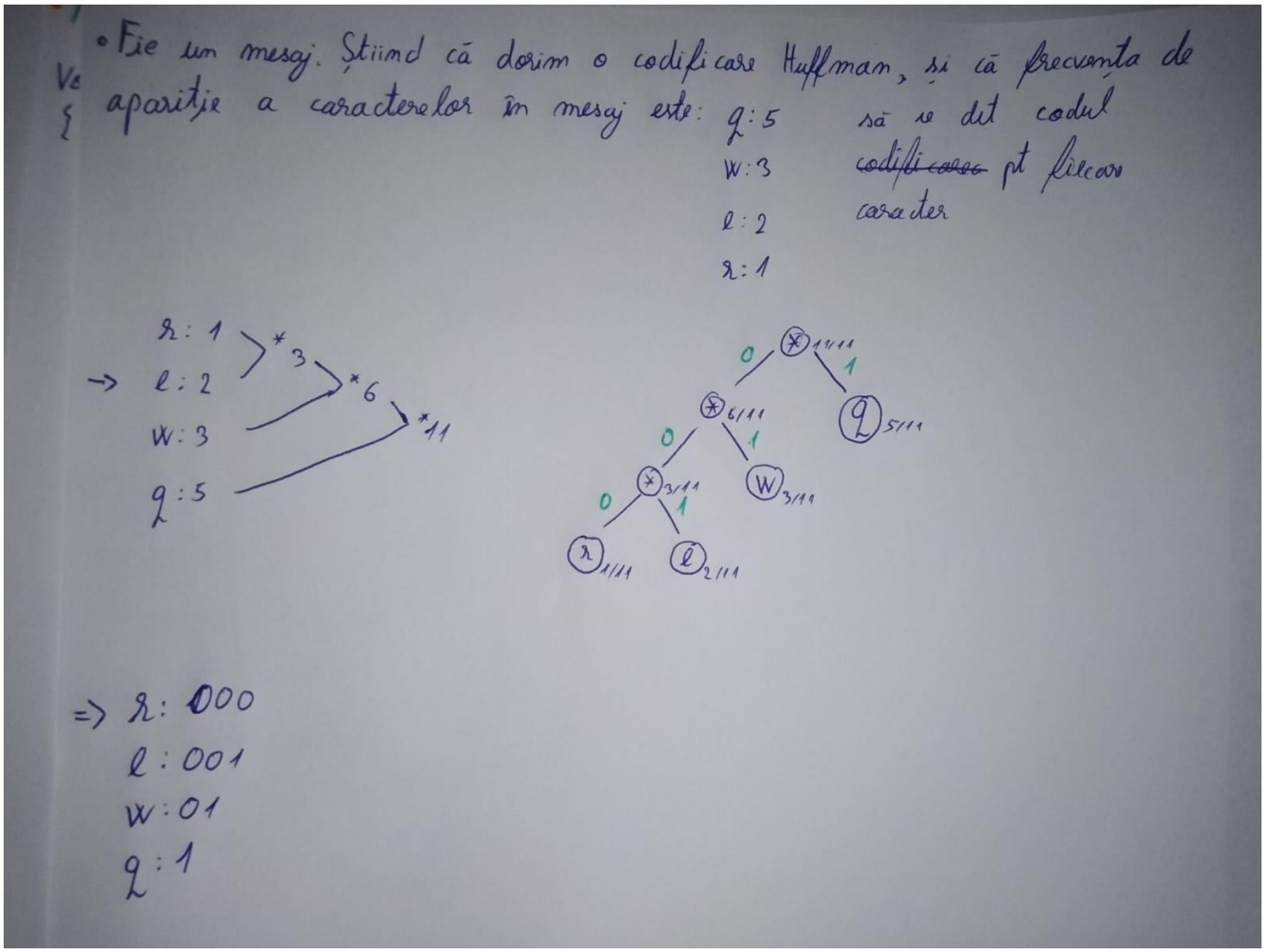
```
struct student
{
	char nume[10];
	char prenume[10];
	float nota;
	struct student *st,*dr;
}*rad;

void afis(struct student *r){
	if(r->nota >=7 && r->nota<8) //daca nodul are intre 7 si 8, asta inseamna ca si pe st si pe dr pot fi note intre 7 si 8
	{
		printf("%s%s\n",r->nume, r->prenume);
		afis(r->st);
		afis(r->dr);
	}
	else
		if(r->nota<7) //dc nodul are nota mai mica ca 7→o nota intre 7 si 8 este pe partea dr
			afis(r->dr)
	else
		if(r->nota>=8) //dc stud are nota mai mare ca 8→o nota intre 7 si 8 este pe partea st
			afis(r->st)

}
```

Problema 9:

Se da un graf in care nodurile reprezinta localitati, iar arcele reprezinta existenta drumurilor asfaltate intre aceste localitati. Acest graf este neconex, insemnand ca exista grupuri de localitati intre care nu se poate ajunge folosind doar drumuri asfaltate. Sa se proiecteze un algoritm care gaseste intre care localitati din grupuri neconectate este eficient de investit in asfaltarea unui drum, astfel incat, la terminarea lucrarilor, drumul intre oricare 2 orase care initial nu erau conectate localitati, sa fie de distanta minima. Distanta drumului care trebuie proiectet se considera 0.

Problema 10:

Fie un mesaj. Stiind ca dorim o codificare Huffmann, si ca frecventa de aparitie a caracterelor in mesajul necodificat este: q 5, w 3, e 2, r 1, sa se determine codul pentru fiecare caracter.



Problema 11:

Pentru o serie de studenti se cunosc: numele, prenumele, nota la PAA. Considerand ca pentru stocarea datelor despre studenti se foloseste o structura de tip arbore binar ordonat, sa se scrie o functie care determina predecesorul unui anumit nod dat, fara a genera secventa nodurilor in inordine.

```
struct student
{
        char nume[10];
        char prenume[10];
        float nota;
        struct student *st,*dr;
}*rad;

Struct student *predecesor(struct student *t){
        t=t->st;
        while(t->dr!=NULL)
                t=t->dr;
        return t;
}


Struct student *succesor(struct student *t){
        t=t->dr;
        while(t->st!=NULL)
                t=t->st;
        return t;
}
```

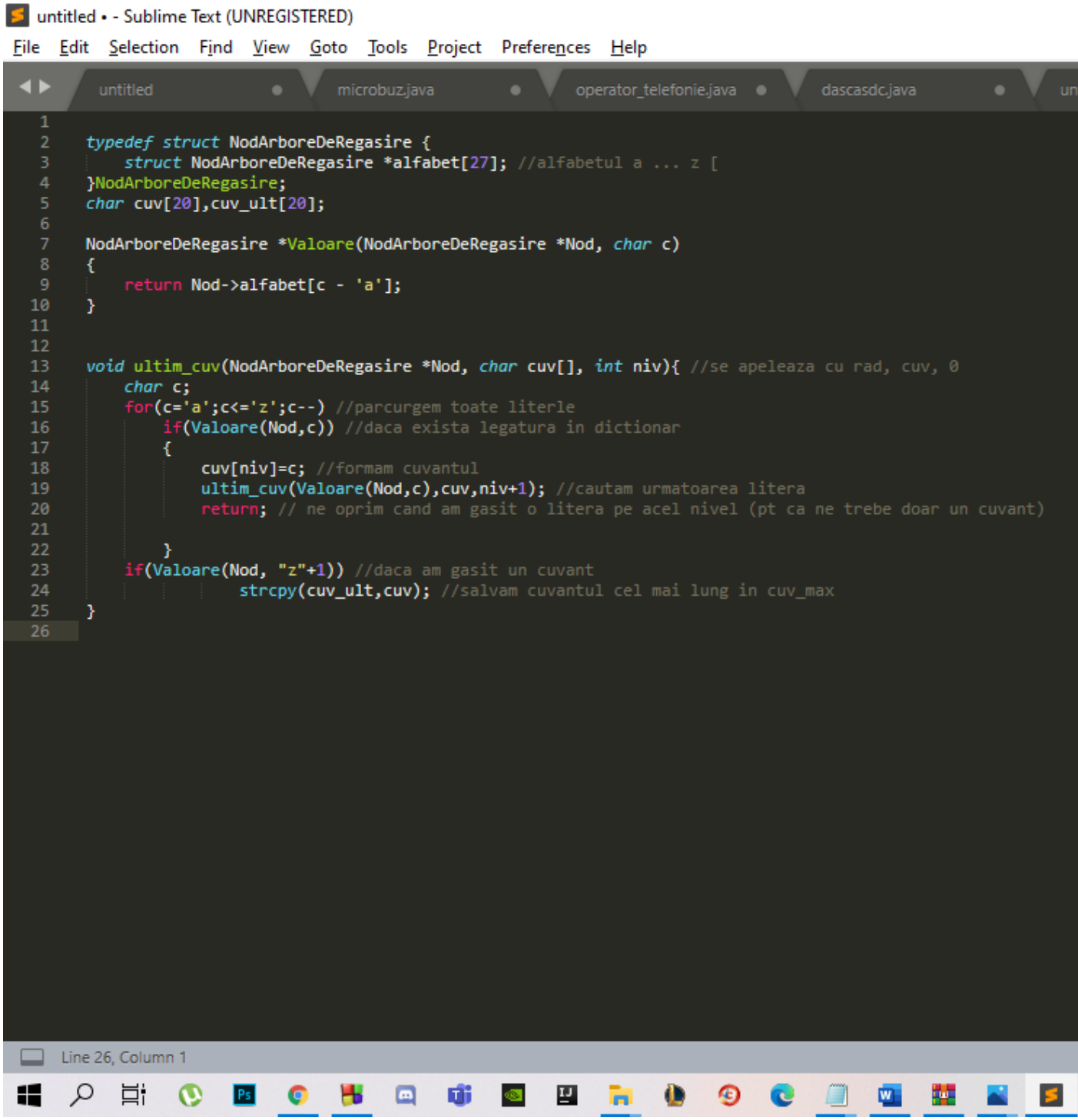
Problema 12:

Pentru o serie de studenti se cunosc: numele, prenumele si anul nasterii. Considerand ca pentru stocarea datelor despre studenti se foloseste o structura de tip arbore binar ordonat, sa se scrie o functie pentru afisarea celui mai tanar student.

```
struct student
{
	char nume[10];
	char prenume[10];
	int an;
	struct student *st,*dr;
}*rad;

void afis_cel_mai_tanar(struct student *r){
	while(r->dr!=NULL)
		r=r->dr //cel mai mare an --> cel mai tanar student
	printf("%s%s\n", r->nume, r->prenume);
}
```

Problema 13:

Fie un dictionar implementat cu ajutorul unui arbore de regasire. Sa se scrie o functie care efectueaza afisarea primului cuvant din dictionar.



Problema 14:

Pentru angajatii unei firme se cunosc ID-ul (intreg) si anul angajarii. Stiind ca un angajat poate avea mai multi subalterni directi, dar un singur sef direct, angajatii se stocheaza intr-un arbore generalizat dat prin tablouri de primul fiu si frate drept. Sa se scrie o functie pentru afisarea tuturor subalternilor unui angajat dat ca parametru prin ID.

Problema 15:

Fie o retea de localitati (L1, L2, ..., LN) pentru care se cunosc conexiunile directe intre localitati si timpul de parcurgere a distantei intre oricare doua localitati interconectate prin drum direct. O firma de transport cu sediul in localitatea L1 doreste sa determine traseele cele mai rapide prin care poate ajunge din localitatea L1 in fiecare din localitatile ramase (in total N-1 trasee). Stiind ca in fiecare localitate trebuie sa stationeze un numar de M minute.

Problema 16:

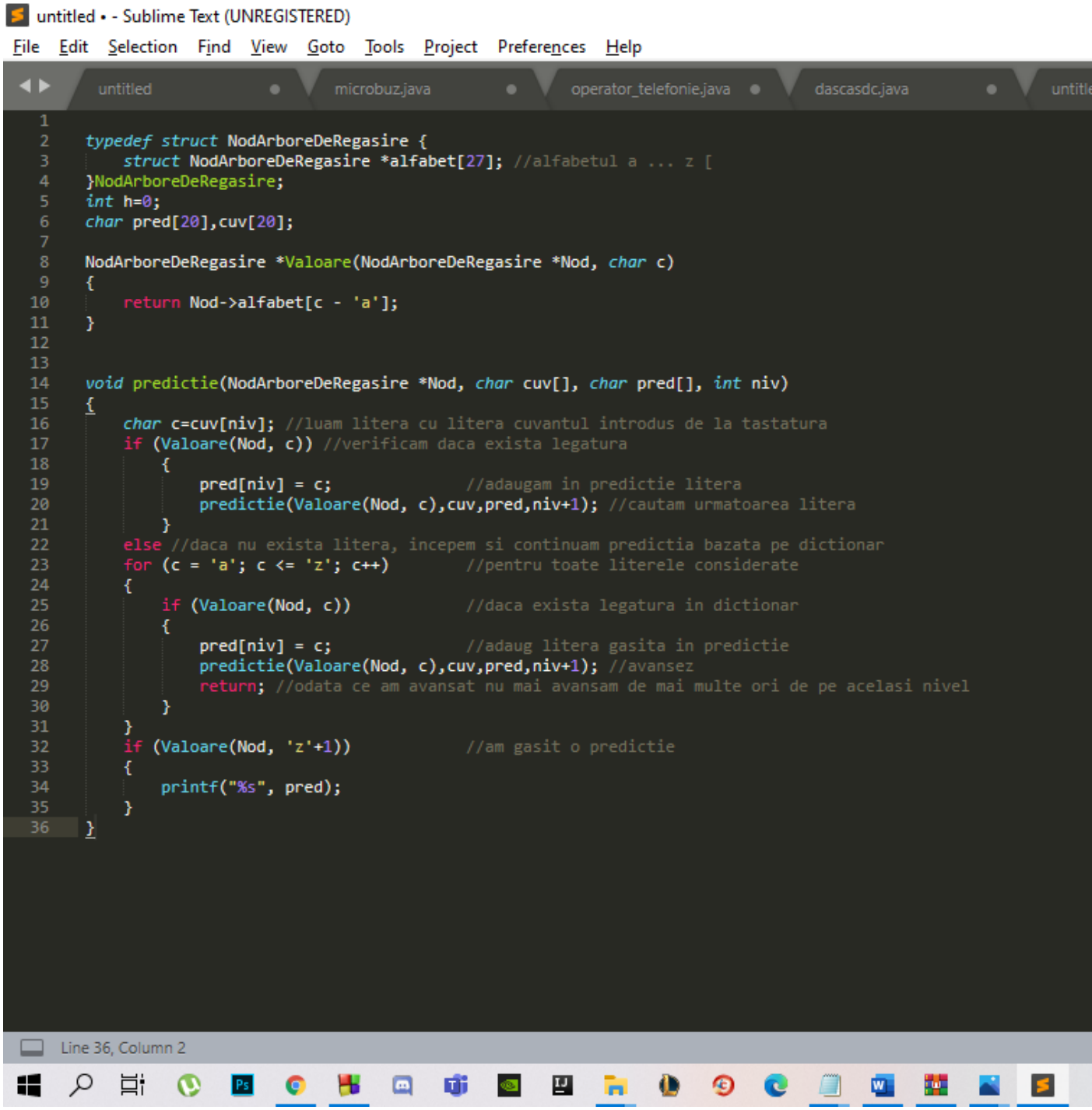
Se da un arbore cu N noduri, prin tabloul cu indicatori spre parinte. Sa se scrie o functie care primeste ca parametru un intreg reprezentand un nivel din arbore, apoi determina si afiseaza cate noduri se afla pe acel nivel.

Problema 17:

Fie un dictionar implementat cu ajutorul unui arbore de regasire. Sa se scrie o functie care primeste ca parametru un cuvant si efectueaza afisarea unei sugestii de corectare a ortografiei cuvantului, daca acesta nu se gaseste in dictionar si daca din cuvant lipseste o litera.

Exemplu:

Dictionarul contine cuvantul „computer”, dar cuvantul primit ca parametru de functie este „computr”.




```c
typedef struct NodArboreDeRegasire {
    struct NodArboreDeRegasire *alfabet[27]; //alfabetul a ... z [
}NodArboreDeRegasire;
int h=0;
char pred[20],cuv[20];

NodArboreDeRegasire *Valoare(NodArboreDeRegasire *Nod, char c)
{
    return Nod->alfabet[c - 'a'];
}


void predictie(NodArboreDeRegasire *Nod, char cuv[], char pred[], int niv)
{
    char c=cuv[niv]; //luam litera cu litera cuvantul introdus de la tastatura
    if (Valoare(Nod, c)) //verificam daca exista legatura
        {
            pred[niv] = c;              //adaugam in predictie litera
            predictie(Valoare(Nod, c),cuv,pred,niv+1); //cautam urmatoarea litera
        }
    else //daca nu exista litera, incepem si continuam predictia bazata pe dictionar
    for (c = 'a'; c <= 'z'; c++)        //pentru toate literele considerate
    {
        if (Valoare(Nod, c))            //daca exista legatura in dictionar
        {
            pred[niv] = c;              //adaug litera gasita in predictie
            predictie(Valoare(Nod, c),cuv,pred,niv+1); //avansez
            return; //odata ce am avansat nu mai avansam de mai multe ori de pe acelasi nivel
        }
    }
    if (Valoare(Nod, 'z'+1))            //am gasit o predictie
    {
        printf("%s", pred);
    }
}
```

Problema 18:

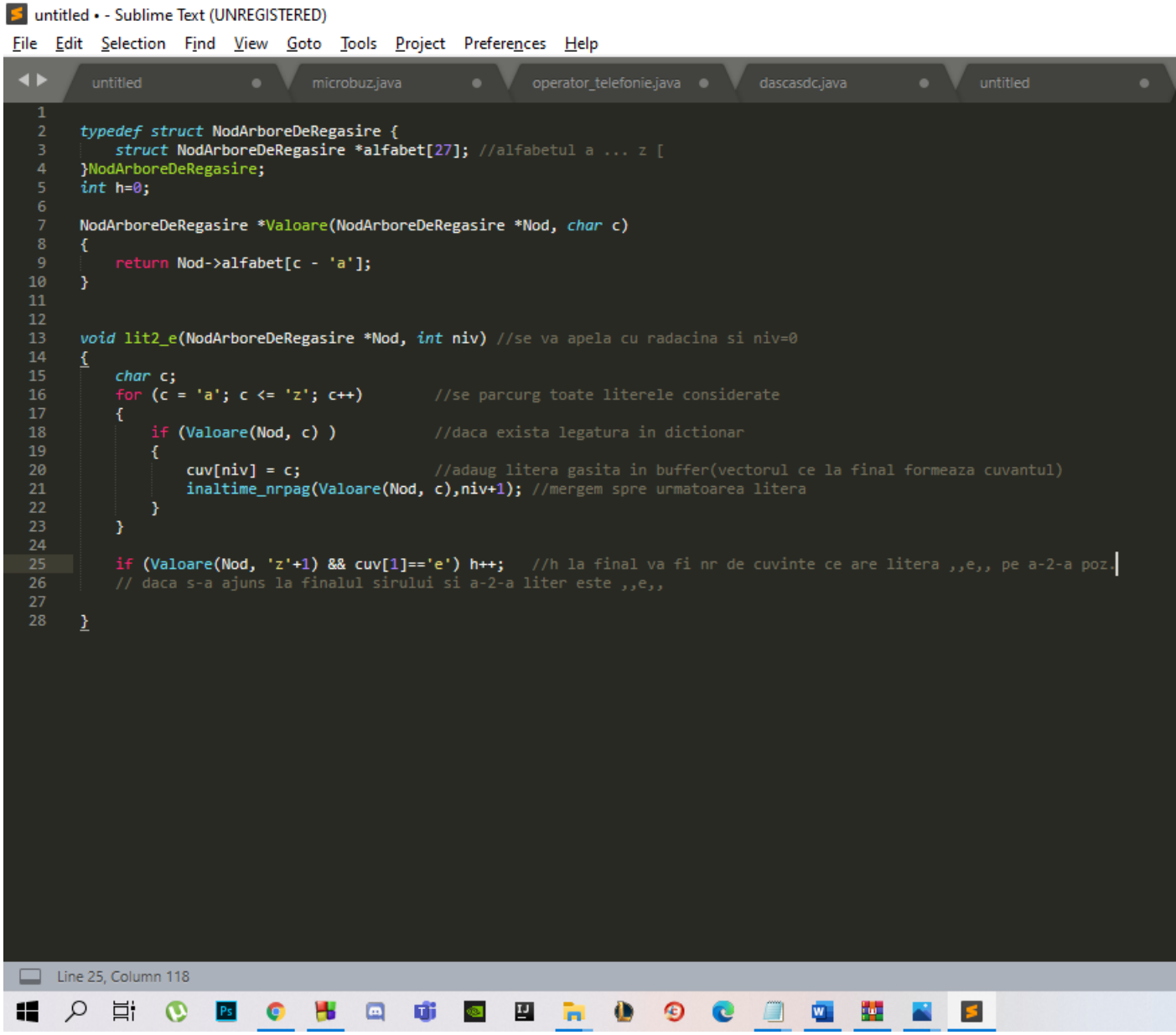
Sa se scrie o funtie care determina complementul unui graf neorientat si neponderat, dat printr-o matrice de adiacenta.
Complement = acel graf cu proprietatea ca fiecare arc din graful initial este sters in graful complementar si fiecare arc lipsa din graful initial exista in cel complementar.

Problema 19:

Se da un graf neorientat ponderat conex cu n varfuri si m muchii in care fiecare muchie are asociat un cost, numar natural strict pozitiv. Folosing algoritmul lui prim determinati un arbore.

Problema 20:

Fie un dictionar implementat cu ajutorul unui arbore de regasire. Sa se scrie o functie care determina numarul cuvintelor care au a doua litera E.



Problema 21:

O structura de tip graf reprezinta prin noduri localitati. Între noduri exista arce (orientate) a căror pondere reprezinta numarul de persoane care au calatorit dintr-o localitate în alta. Stiind ca statistic 1 din 3 călători e infectat cu Coronavirusul Chinez din Wuhan (SARS-CoV2), sa se gaseasca cele doua localitati între care exista cel mai periculos drum, pentru a fi supravegheat:
- Sa se descrie structura de date folosită
- Sa se descrie si explice folosirea algoritmului

Problema 22:

Se considera un graf turneu cu n varfuri avand arc (i, j) intre oricare doua varfuri i, j cu proprietatea ca i este mai mic decat j. Afisati toate drumurile elementare de la un varf x la un varf y, x si y citite, avand proprietatea ca x este mai mic decat y.

Problema 23:

Se da harta unei regiuni, reprezentata prin matricea de adiacenta a unui graf neorientat, ponderat, nodurile reprezentand orase din regiune si ponderile reprezentand distantele dintre orase. Sa se scrie un program care determina numarul minim de stalpi care trebuie folositi de o companie de telecomunicatii care conecteaza toate orasele, stiind ca distanta dintre 2 stalpi este de 100 m.