Napisati program za formiranje binarnog stabla koje sadrži izraz sa 4 osnovne operacije(+, -, *, /), pri čemu su prisutne sve zagrade, tj. (a op b)

Vrednost izraza

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

struct drvo{
  union {
        int broj;
        char oper;
  } sadrzaj;
  struct drvo *levi,*desni;};

struct stek {
  struct drvo *sadrzaj;
  struct stek *rep;
};
#define novis(x) x=(struct stek *)malloc(sizeof(struct stek))
#define novid(x) x=(struct drvo *)malloc(sizeof(struct drvo))
```

Vrednost izraza

```
struct stek* push(struct stek *p,struct drvo *d){
    struct stek *temp;
    novis(temp);
    if(!temp) exit(0);
    temp->sadrzaj=d;   temp->rep=p;
    p=temp;
    return p;
}
struct drvo* pop(struct stek **p){
   struct stek *temp;
   struct drvo *d;
   temp=*p;  *p=(*p)->rep;
   d=temp->sadrzaj;
   free(temp);
   return d;
}
int isoper(char c){
    if((c=='+') || (c=='-') || (c=='*') || (c=='/')) return 1;
    return 0;
}
```

Vrednost izraza

```
struct drvo* napravi(){
  char s[50],ch,op;
  int br=0;
  struct drvo *temp,*l,*d,*o;
  struct stek *poc=NULL;
  while((ch=getchar())!='\n'){
        if (isdigit(ch)){
              while(isdigit(ch)) {
                    s[br++]=ch;
                    ch=getchar();
              }
              s[br]='\0';
              br=0;
              novid(temp);
              if(!temp) exit(0);
              temp->sadrzaj.broj=atoi(s);
              temp->levi=temp->desni=NULL;
              poc=push(poc,temp);
        }
...
```

Vrednost izraza

```
...
        if(isoper(ch)) {
                novid(temp);
                if(!temp) exit(0);
                temp->sadrzaj.oper=ch;
                temp->levi=temp->desni=NULL;
                poc=push(poc,temp);
        }
        if(ch==')'){
                d=pop(&poc);
                o=pop(&poc);
                l=pop(&poc);
                o->levi=l;
                o->desni=d;
                poc=push(poc,o);
        }
  }
  return poc->sadrzaj;
}
```

Vrednost izraza

```
int vrednost(struct drvo *p){
  int r,d;
  if((p->levi) && (p->desni)){
     switch(p->sadrzaj.oper){
       case '+' : r=vrednost(p->levi)+vrednost(p->desni); break;
       case '-' : r=vrednost(p->levi)-vrednost(p->desni); break;
       case '*' : r=vrednost(p->levi)*vrednost(p->desni); break;
       case '/' : d=vrednost(p->desni);
                        if (d) r=vrednost(p->levi)/d;
                        else exit(0);
                        break;
     }
  }
  else r=p->sadrzaj.broj;
  return r;
}
```

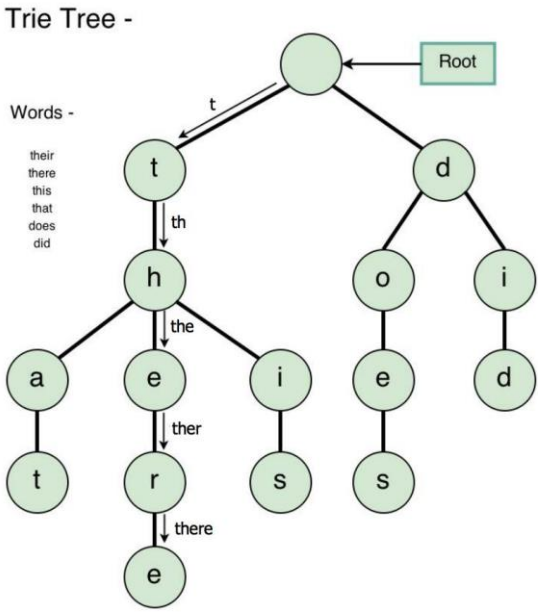
Vrednost izraza

```
void ispis(struct drvo *p){
  if ((p->levi) && (p->desni)){
       printf("(");
       ispis(p->levi);
       printf(" %c ",p->sadrzaj.oper);
       ispis(p->desni);
       printf(")");
  }
  else printf("%d",p->sadrzaj.broj);
}

main(){
  struct drvo *exp;
  exp=napravi();
  ispis(exp);
  printf("\nVrednost je %d\n",vrednost(exp));
}
```

Na ulazu se zadaje ceo broj N, a zatim N reči. Formirati **trie** ili **digitalno stablo** čiji su elementi strukture koje mogu da sadrže jedno slovo engleskog alfabeta (nije neophodno) i niz pokazivača ka mogućim ostalim čvorovima. Smatrati da ima 26 slova. Ovo stablo je pogodno za čuvanje teksta. Primer je prikazan na slici.

Napisati funkciju koja za datu reč sa ulaza proverava da li se nalazi u stablu.



Digitalno stablo

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>


typedef struct drvo
{
  char c;

  struct drvo *sledeci[26];
}drvo;


int mesto(char c)
{
  int m = c - 97;
  return m;
}
```

Digitalno stablo

```
void dodaj(drvo *koren)
{
   char rec[50];
   int t, i, j = 0;
   drvo *pom = koren;

   scanf("%s", rec);
   int n = strlen(rec);

   for (i=0; i<n; i++)
   {
         t = mesto(rec[i]);

         if (pom->sledeci[t] == NULL)
         {
                  drvo *novi = (drvo*)malloc(sizeof(drvo));
                  novi->c = rec[i];
                  pom->sledeci[t] = novi;
                  for (j=0; j<26; j++)
                           novi->sledeci[j] = NULL;
                  pom = novi;
         }
         else
         {
                  pom = pom->sledeci[t];
         }
   }
}
```

Digitalno stablo

```
void da_li_je_rec_u_stablu(drvo *koren)
{
   char rec[50];

   scanf("%s", rec);

   int n = strlen(rec);
   int i,k;

   for(i=0; i<n; i++)
   {
         k = mesto(rec[i]);

         if ((koren->sledeci[k] == NULL) || (koren->sledeci[k]->c != rec[i]))
         {
                  printf("Nije u stablu! \n");
                  return;
         }
         else
         {
                  koren = koren->sledeci[k];
         }
   }

   printf("Rec je u stablu! \n");
   return;
}
```

Digitalno stablo

```
int main()
{
  drvo *koren = (drvo*)malloc(sizeof(drvo));
  koren->c = '1';
  int i,n;

  for (i=0; i<26; i++)
       koren->sledeci[i] = NULL;

  scanf("%d", &n);

  for(i=0; i<n; i++)
       dodaj(koren);

  printf("Uneti rec za proveru: \n");

  da_li_je_rec_u_stablu(koren);

}
```