```
#define max 20
#define NULL 0

int T[max]; /* T[i] sera le nombre d'arbres de taille i*/

void calcT()
{
      int i;
      T[0]=1;
      for (i=1;i<=max;i++)
      {int j, /*taille de sous-arbre gauche*/
            somme=0;
            for (j=0;j<i;j++)
                  somme+=T[j]*T[i-1-j];
            T[i]=somme;
            printf("T[%i]=%i\n",i,T[i]);
      }
}

struct arbre
{
      struct arbre *g,*d;
};

int taille (struct arbre *racine)
{
      return racine==NULL ? 0 : 1+taille(racine->g)+taille(racine->d);
}

int code (struct arbre *racine)
{
      int i, somme=0;
      if (racine == NULL) return 0;
      for (i=0;i<taille(racine->g);i++)
            somme += T[i]*T[taille(racine)-1-i];
      somme += code(racine->g)*T[taille(racine->d)];
      return somme+code(racine->d);
}

struct arbre *decode(int n, int c)
      /* n la taille de l'arbre et c le code*/
{
      struct arbre *racine;
      int i=-1;
      if (n==0) return NULL;
      racine = malloc(sizeof (struct arbre));
```

```
        do {i++;c-=T[i]*T[n-1-i];} while (c>=0);
        c+=T[i]*T[n-1-i]; /*i est la taille du sous-arbre gauche*/
        racine->g=decode(i,c/T[n-1-i]);
        racine->d=decode(n-1-i,c%T[n-1-i]);
        return racine;
}
```