complexité
Examen

Nous nous proposons d'évaluer la complexité du tri par tas (*Heapsort*). Un tas est un tableau considéré comme un arbre binaire "ordonné verticalement" : l'élément de rang *i* a deux fils aux rangs *2*i* et *2*i+1* (il peut évidemment en avoir un seul ou aucun). Il est ordonné verticalement car la valeur de l'élément de rang *i* est supérieure à la valeur de ses deux fils, pour tout *i*. Dans la définition précédente on suppose que l'indice du premier élément du tableau est 1.

Question 1

Dessiner la représentation sous forme d'arbre du tas :

10	6	9	4	3	8	7

Question 2

Nous supposons que la partie du tableau qui va de *1* à *n* est organisée en tas, sauf le premier élément qui n'est pas à sa place. La fonction *descendre* range l'élément de rang *1* à sa place dans le tas :

```
void descendre( int t [], int n, int nMax){
    int e1, e2, em;
    if(n*2>nMax) return;
    else {
       e1 = 2*n;
       e2 = 2*n+1;
       if(e2>nMax) em= e1; else
         if (t[e1-1] > t[e2-1]) em = e1; else em = e2;
       if( t[n-1] > t[em-1])return;
       else{
          echanger( t[n-1],  t[em - 1]);
          descendre( t, em, nMax);
       }
   }
}
```

Dans l'exemple de la question 1, remplacer le premier élément par 2, et appliquer l'algorithme précédent ; dessiner l'arbre obtenu.

Evaluer le nombre d'appels de la fonction *descendre(t, 1, n)* en fonction de *n* dans le cas où $n=2^m-1$ et dans le pire cas.

Question 3

La fonction *organiserTas* définie ci-dessous permet d'organiser un tableau quelconque en un tas :

```
void organiserTas( int t [], int n ){
    for(int i = n/2; i>=1; --i) descendre(t, i, n);
}
```

Nous nous proposons de calculer la complexité de la fonction *organiserTas* en comptant le nombre maximum de déplacements d'éléments réalisés lors de l'organisation en tas :

- n/4 éléments vont descendre au maximum de 1 place.
- n/8 éléments vont descendre au maximum de 2 places.
- ...
- 1 élément va descendre au maximum de $\log_2(n)$ places.

Si $n = 2^m-1$ le nombre de déplacements est donc $\frac{n}{2} \times \sum_{i=1}^{\log_2(n)} i/2^i$

Calculer la récurrence : $C_0 = 0$, $C_i = C_{i-1} + \frac{i}{2^i}$ On posera $Z_i = 2^i \times C_i$, et on montrera $Z_i = 2^{i+1} - i - 2$

En déduire le nombre maximum de déplacements lors de l'organisation d'un tableau en tas.

Question 4

Le tri par tas est défini de la façon suivante :

```
void trierTas( int t [], int n ){
     organiserTas( t, n);
     for( int i = n; i>1; --i ){
        echanger(t[0], t[i-1]);
        descendre(t, 1, i-1);
     }
}
```

Lors du tri, 2 éléments vont descendre au maximum de 1 place , 4 éléments vont descendre au maximum de 2 places, ... 2^i éléments vont descendre au maximum de i places. Supposons que *m* soit la première puissance de 2 supérieure ou égale à *n*.

La récurrence suivante donne une borne supérieure au nombre maximum de déplacements, lors d'un tri :

$C_0 = 0$, $C_m = C_{m-1} + m \times 2^m$ Calculer C_m en posant $Y_m = \frac{C_m}{2^m}$