❑ **Exercice** : *type A*

On rappelle que le principe de la compression LZW est d'attribuer un code aux préfixes rencontrés lors de la lecture du texte à compresser de façon à disposer d'un code compact si ce code se présente à nouveau. Initialement, seuls les codes de l'alphabet (usuellement les caractères ASCII) sont présents dans la table de codage.

Ici, on veut compresser le texte `saisissais` sur l'alphabet `{a,i,s}`, par souci de simplicité on attribue initialement les codes `a` $\to 0$, `i` $\to 1$ et `s` $\to 2$. Le début de l'algorithme va donc consister à attribuer un nouveau code au premier préfixe non encore codé qui apparaît lors de la lecture du texte. Et donc, ici, on attribue le code `3` au prefixe `sa` et on emmettra le code de `s`.

1. Poursuivre le déroulement de cet algorithme en complétant le tableau suivant :

Position dans le texte	Code émis	Nouveau préfixe ajouté
`saisissais`	2	`sa` $\to 3$

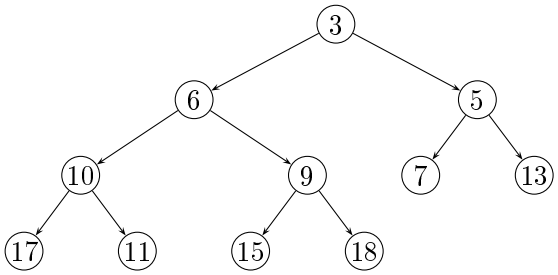
2. Quel est le taux de compression obtenu en supposant qu'un octet permet de représenter chaque code ?

3. Décompresser le texte T codé par suite de codes `[0; 1; 4; 6; 5; 2; 3; 2; ]` sur l'alphabet `{l, a, e, r}` avec les codes `l` $\to 0$, `a` $\to 1$, `e` $\to 2$ et `r` $\to 3$ en expliquant comment est reconstruit le dictionnaire de décompression.

4. Rappeler rapidement le principe de l'algorithme de compression de Huffman et compresser le texte T à l'aide de cet algorithme.

❑ **Exercice** : *type B*

1. Rappeler la défintion d'un tas binaire (min)
2. On suppose qu'un tas est représenté par un tableau $t = (t_0, \dots t_{n-1})$. Lorsqu'ils existent quels sont les indices des fils de t_i ?
3. Quel est l'indice (lorsqu'il existe) du père de t_i ?
4. Vérifier que l'arbre binaire suivant possède bien la structure de tas et donner sa représentation sous forme de tableau :



5. Rappeler le principe d'insertion d'un nouvel élément dans un tas binaire puis détailler les étapes de l'insertion de 4 dans le tas représenté à la question 4.
6. Rappeler le principe d'extraction du minimum d'un tas binaire, puis détailler les étapes de l'extraction du minimum du tas représenté à la question 4.
7. Dans la suite, on utilisera la structure de données suivante permettant de représenter un tas en OCaml :

```
type 'a heap = {mutable size : int; data : 'a array};;
```

Quel est le rôle du champ `size` ? Pourquoi est-il déclaré en `mutable` ?

8. Ecrire une fonction qui renvoie un tas de taille maximale `1000` et qui contient des valeurs entières.
9. Ecrire la fonction d'insertion d'un élément dans un tas binaire contenant des entiers. On renvoie `true` si l'insertion est possible (c'est à dire que le tas n'est pas plein) et `false` sinon. La signature de cette fonction est donc `insertion: 'a -> 'a heap -> bool`