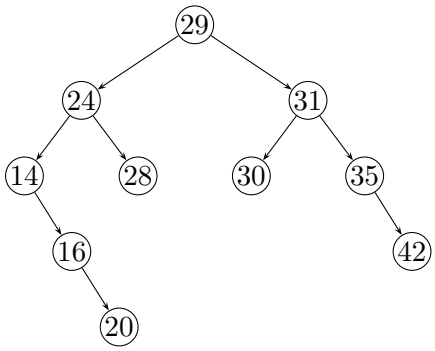
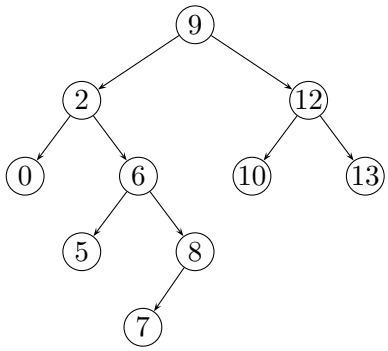
❑ **Exercice 1** : *Arbre binaire de recherche*
Cette exercice est à traiter en OCaml en utilisant le type :

```
type ab = Vide | Noeud of ab * int * ab ;;
```

1. On considère l'arbre binaire suivant :



2. Justifier (en utilisant la définition) qu'il s'agit bien d'un arbre binaire de recherche (ABR).
3. On suppose *seulement dans cette question* que le fils gauche du noeud 16 porte l'étiquette $x \in \mathbb{N}$. Rappeler la caractérisation d'un ABR par son parcours infixe et utiliser cette caractérisation pour déterminer les valeurs possibles de x.
4. Rappeler l'algorithme d'insertion d'un élément dans un arbre binaire de recherche
5. Détailler l'insertion des valeurs 17 et 33 dans cette arbre et dessiner l'arbre obtenu après ces insertions.
6. Donner une implémentation en OCaml de l'algorithme d'insertion sous la forme d'une fonction `insere` de signature `abr -> int -> abr`
7. On note m la valeur minimale d'un ABR. Justifier que l'arbre obtenu en remplaçant le noeud d'étiquette m par son sous arbre droit, est un ABR.
8. Ecrire une fonction `extraire_min` en OCaml de signaure `abr -> int * abr`, qui renvoie un couple composé du minimum d'un arbre binaire de recherche et de cet arbre privé du noeud de valeur minimale. On gère le cas de l'arbre vide avec `failwith`
9. Donner la complexité de `extraire_min`.
10. Afin de supprimer une valeur dans un ABR, on propose de remplacer cette valeur par la plus petite valeur présente dans son sous arbre droit en y supprimant cette valeur grâce à la fonction précédente. Détailler le fonctionnement de cette méthode sur l'arbre suivant d'où on veut supprimer la valeur 2 :



11. Donner une implémentation en OCaml sous la forme d'une fonction `supprime` de signature `abr -> int -> abr`

❑ **Exercice 2** : *Problème du sac à dos en force brute*
Voir l'énoncé en ligne : chapitre 13 exercice 1