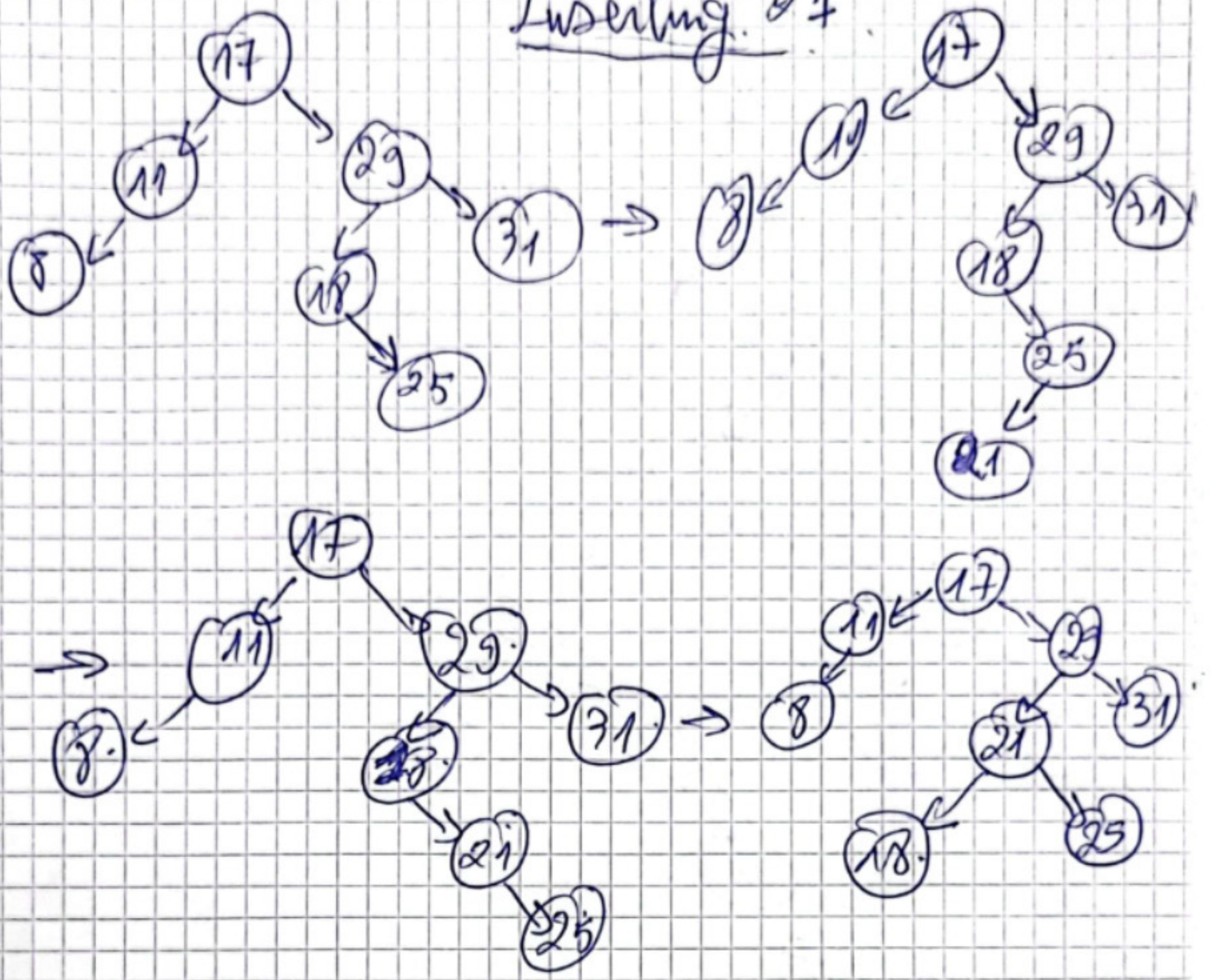
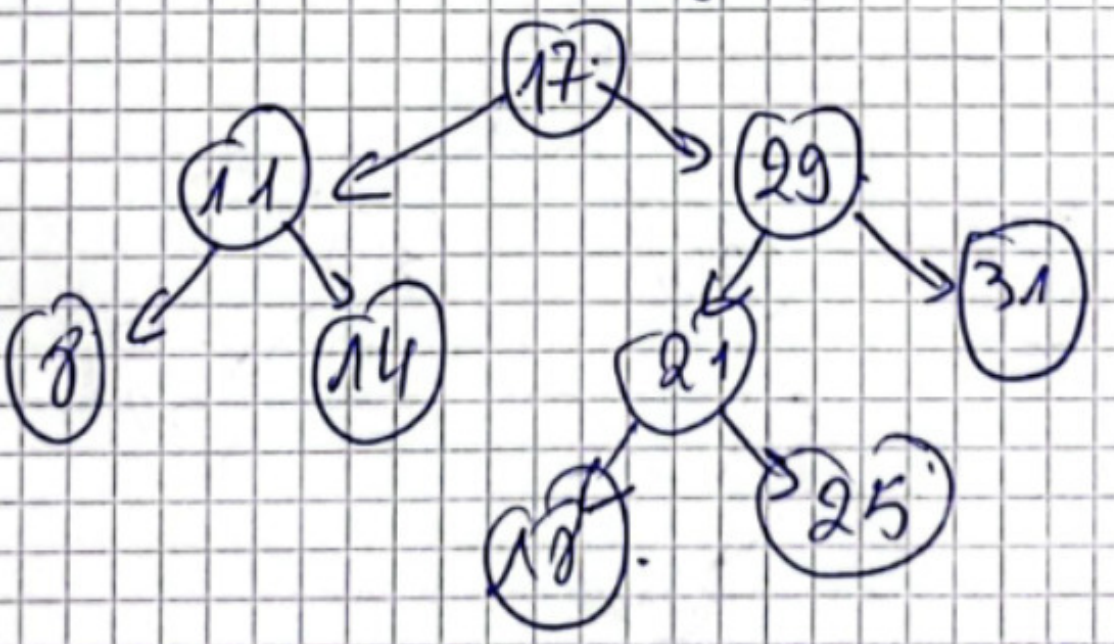
Guechi Ahmed.

Exercice 1 :

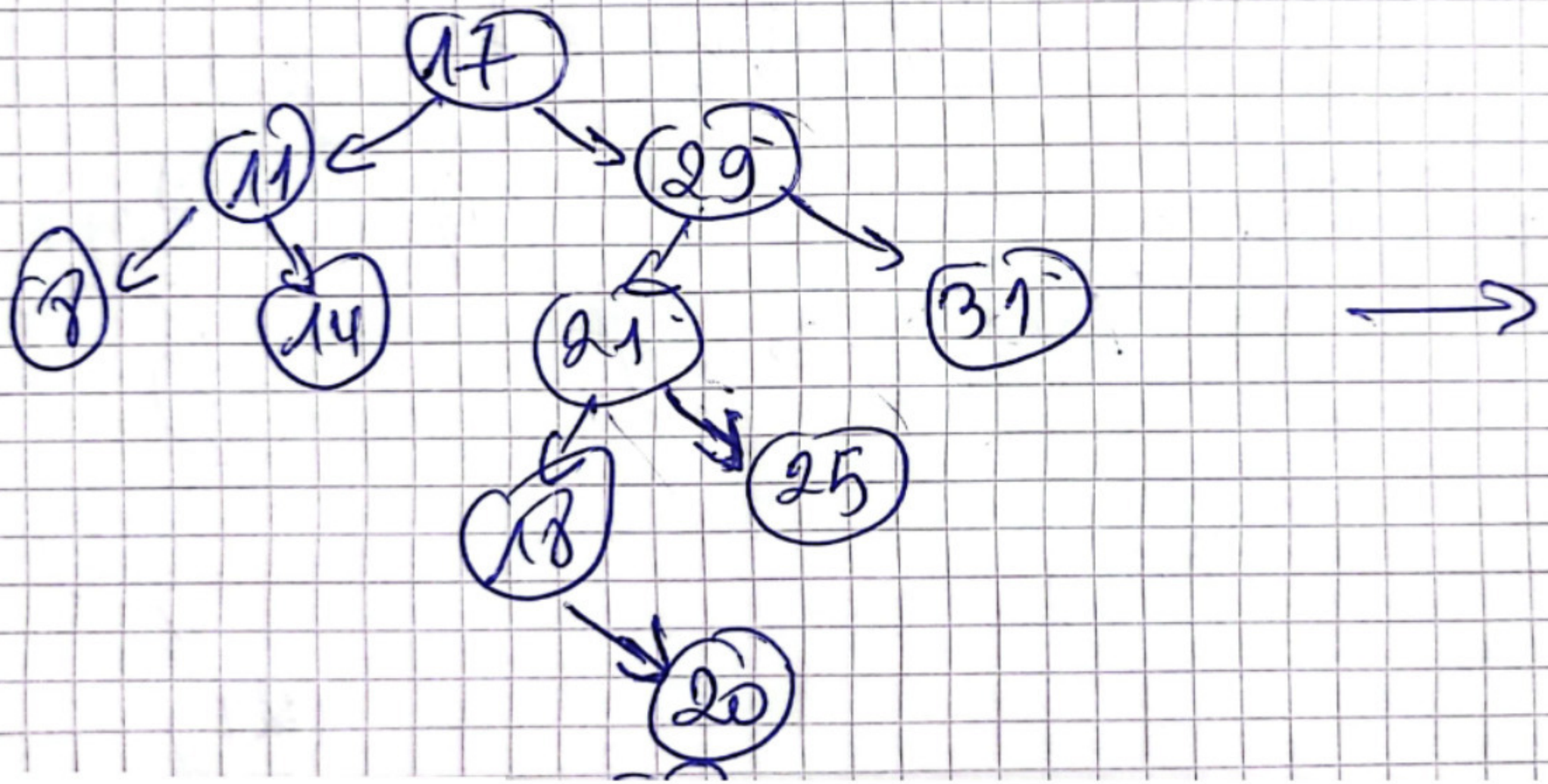
Inserting 21

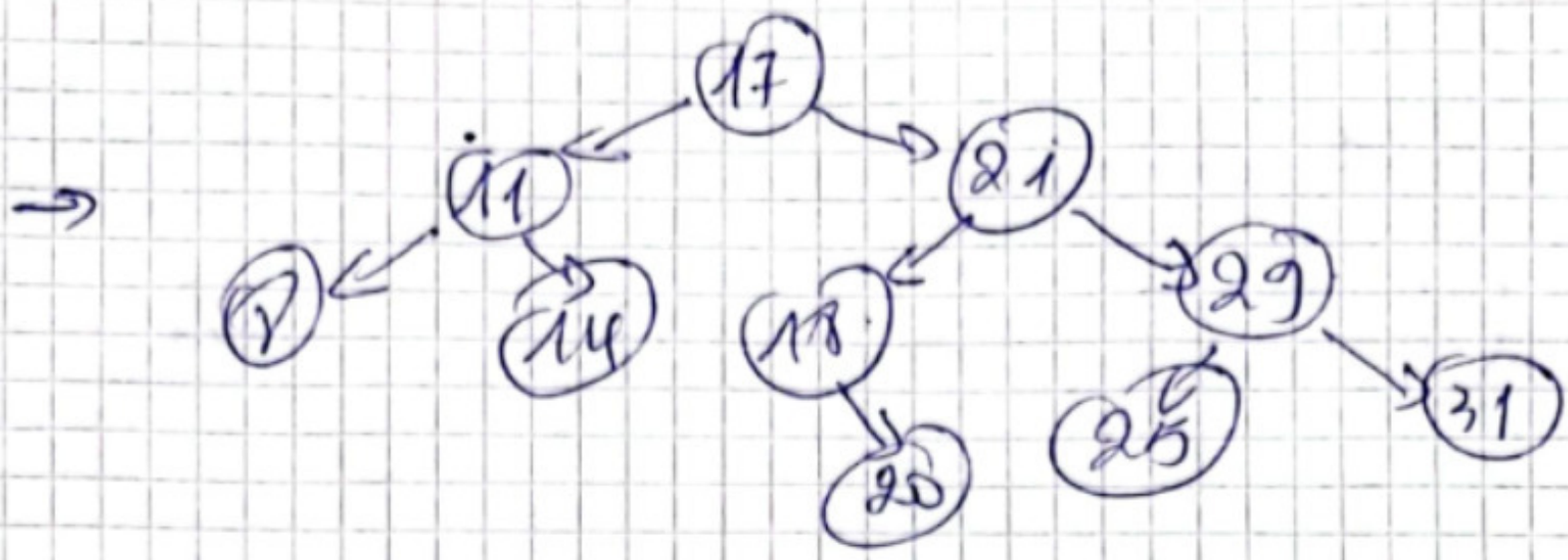


Inserting 14

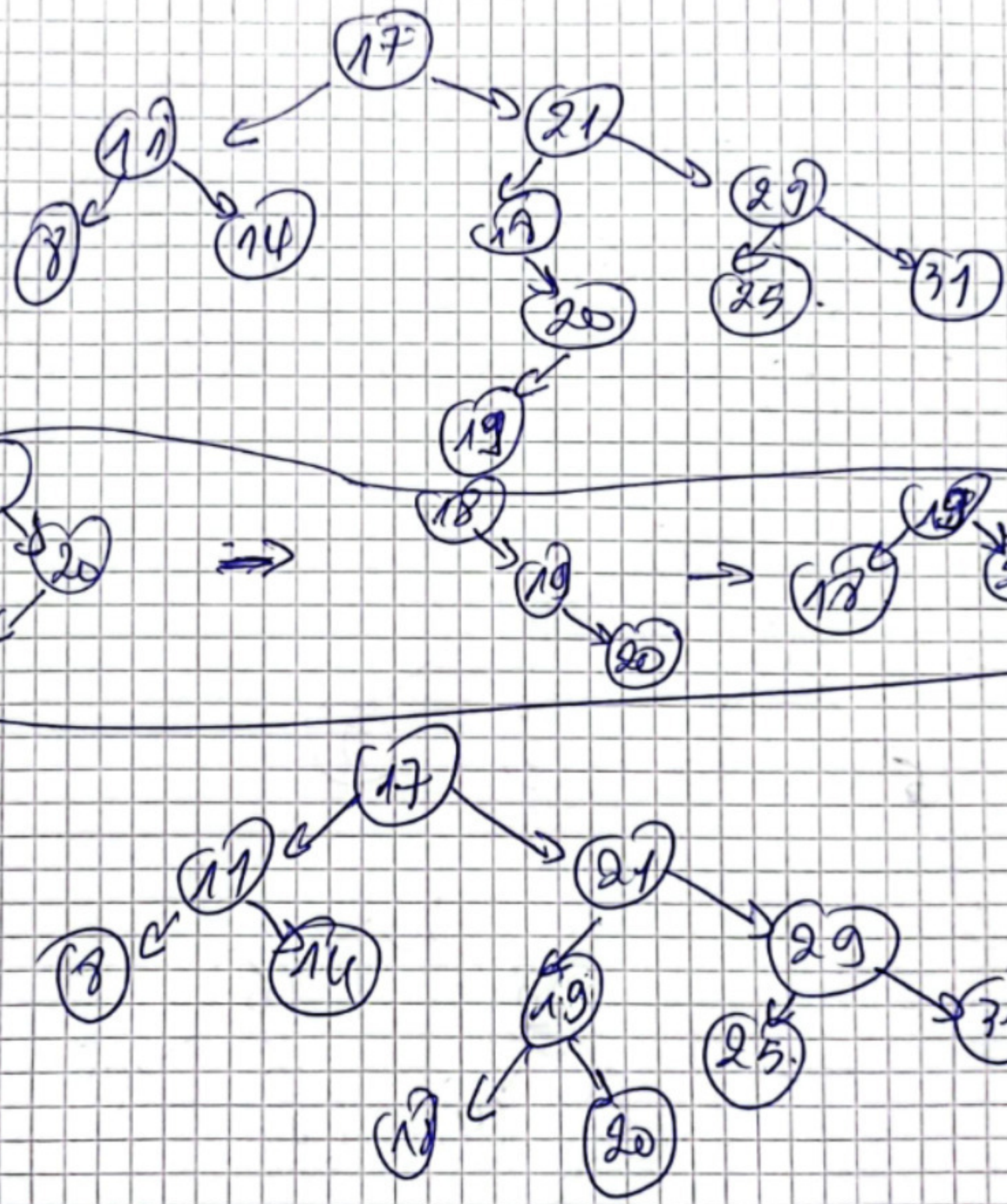


Inserting 20



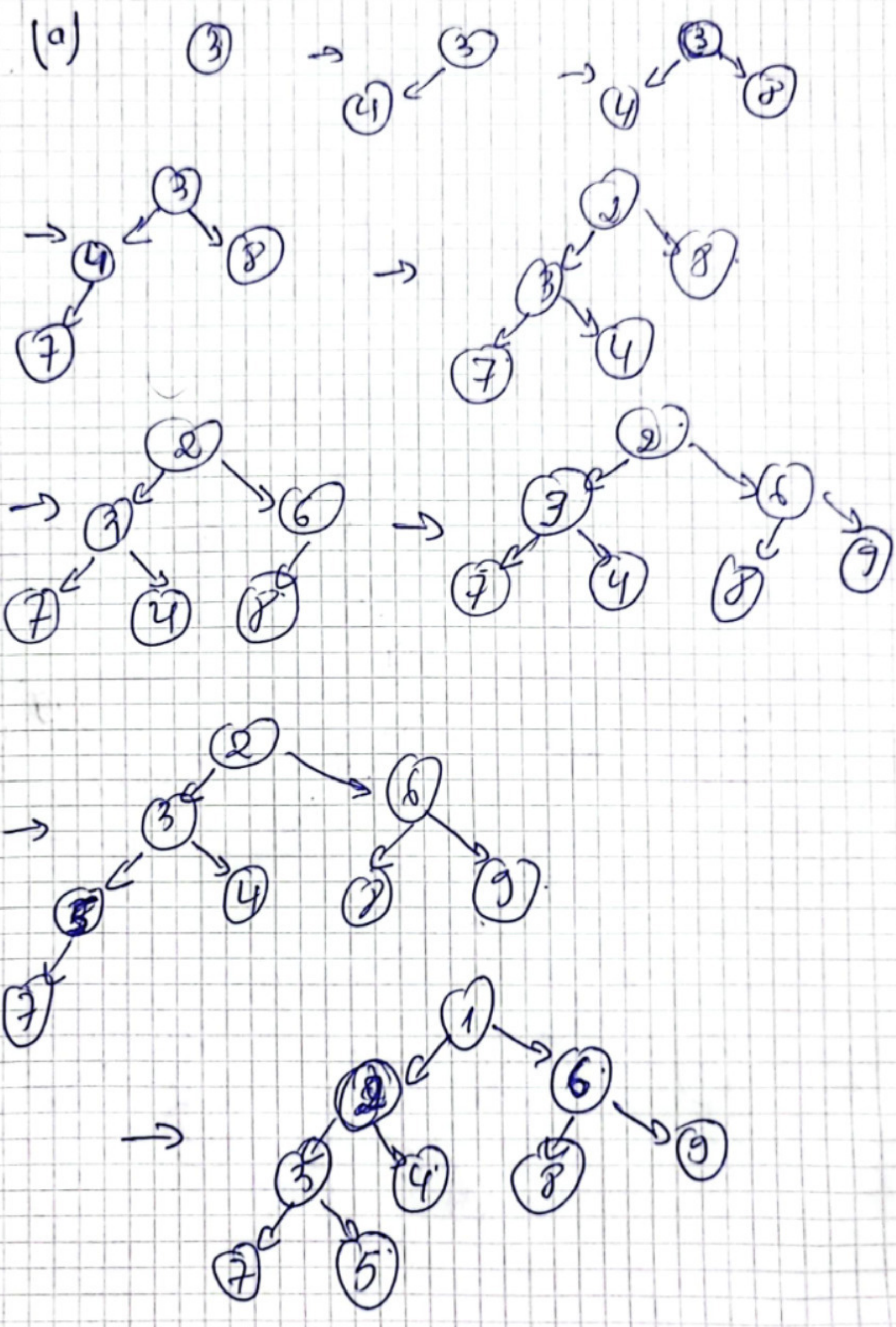


Inserting 19.

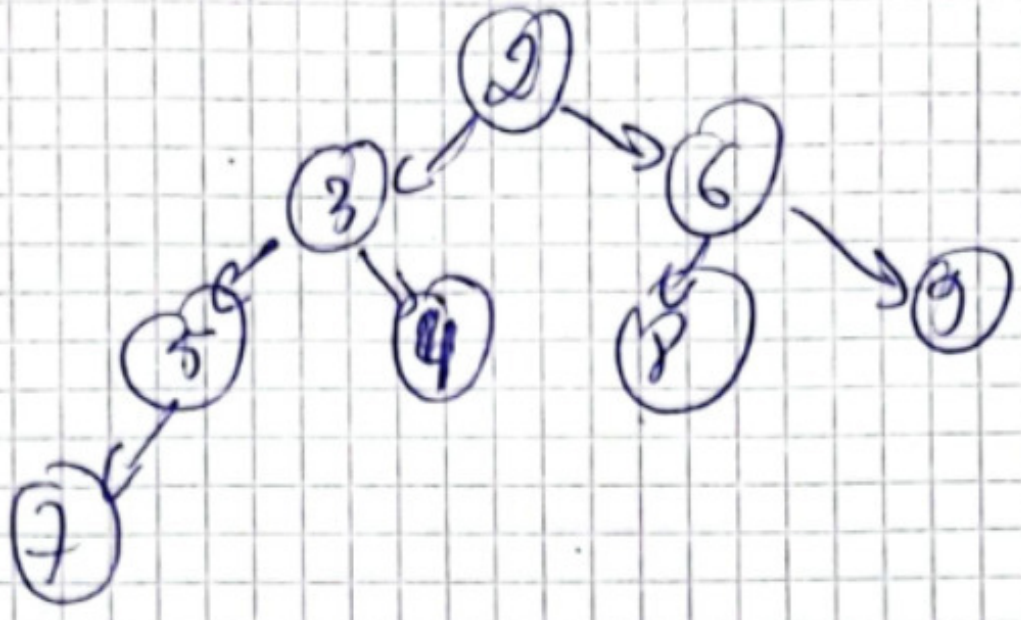


Exercice 2 :

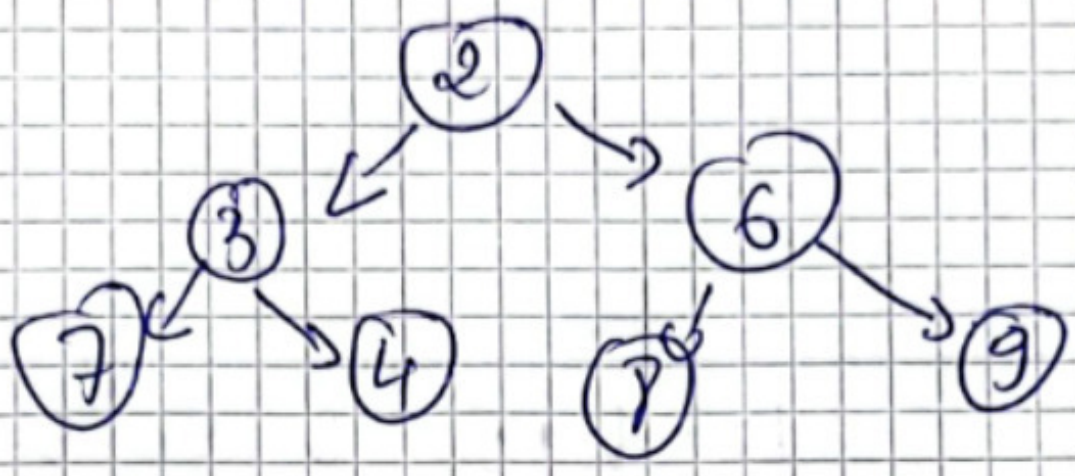
(a)



(b)



(c)



(d) in the obtained tree in question (c) :

- The minimum : 2
- The maximum : 4

* for a min heap of height h :

- The minimum : h
- The maximum : 2h

(f) The minimum of comparision : h

The maximum : 2h

(e) 1) $O(\log N)$

2) $O(N \log N)$

3) $O(N)$

4) $O(\log N)$

Exo 3 :

a/

Key value	Home slot	Prob Sequence
43	1	
23	2	
1	5	
0	3	
15	1	2, 3, 4
31	0	
4	0	1, 2, 3, 4, 5, 6
7	8	
11	9	
3	9	10

Slot	0	1	2	3	4	5	6	7	8	9	10
content	31	43	23	0	15	1	4		7	11	3

b).

Key value	Home Slot	Prob Sequence
43	1	
23	2	
1	5	
0	3	
15	1	2, 4
31	0	
4	0	1, 3, 6
7	8	
11	9	
3	9	10

Slot	0	1	2	3	4	5	6	7	8	9	10
content	31	43	23	0	15	1	4		7	11	3

c)

Key value	Home slot	Prob Sequence
43	1	
23	2	
1	5	
0	3	
15	1	8
31	0	
4	0	4
7	8	4, 0, 7
11	9	
3	9	1, 4, 7, 10

Slot	0	1	2	3	4	5	6	7	8	9	10
Content	31	43	23	0	4	1		7	15	11	3

exercice 4:

1) strcutures :

```
class Book{
   friend ostream& operator<<(ostream&,const Book&);
private:
   string bookName;
   string authorName;
   Book* nextBook;
public:
   Book(string bookN,string authorN):bookName(bookN),authorName(authorN),nextBook(nullptr){};
   ~Book();
   string getbookName()const;
   string getauthorName()const;
   Book* getNextBook();
   void setbookName(string bookN);
   void setauthorName(string authorN);
   void setNextBook(Book* p);
};

class Category{
   friend ostream& operator<<(ostream& out,const Category& a);
   private:
      string CategoryName;
      int NumOfBooks;
      Category* nextCategory;
      Book* books;                  //pointer to the root of the linked list of books
   public:
      Category(string cn):CategoryName(cn),NumOfBooks(0),nextCategory(nullptr),books(nullptr){};
      ~Category();
      Category* getNextCategory()const;
      string getCategoryName()const;
      void setNextCategory(Category* a);
      int getNumberOfBooks()const;
      void addBook(string bn,string an);
      void displayBooks();
};

class Library{
private:
   int NumberOfCat;
   Category* Categories;              // pointer to the root of the linked list of categories
public:
   Library():Categories(nullptr){};
   ~Library(){};
}

2)void addCategory(string CategoryName,int NumOfBooks=0){
      if(Categories==nullptr) Categories=new Category(CategoryName);
      else{
         Category* p=Categories;
         while (p->getNextCategory()!=nullptr){p=p->getNextCategory();}
         p->setNextCategory(new Category(CategoryName));
      }
   };
```

3)
```
void addBookAtBegin(string CategoryName,string bookName,string authorName){
        Category* p=Categories;
        while(p!=nullptr && p->getCategoryName()!=CategoryName){p=p->getNextCategory();}
        if(p==nullptr){
            //if the category doesnt exist, create it
            this->addCategory(CategoryName);
            addBookAtBegin(CategoryName,bookName,authorName);
        }else{
            p->setNumberOfBooks(p->getNumberOfBooks()+1);
            Book* p1=p->getBooks();
            p->setBooks(new Book(bookName,authorName));
            (p->getBooks())->setNextBook(p1);
        }

    };
```
4)
```
void displayBooksInCategory(string CategoryName){
        Category* p=Categories;
        while(p!=nullptr && p->getCategoryName()!=CategoryName){p=p->getNextCategory();}
        if(p==nullptr){
            cout << "This category does not exist(" << CategoryName << ")" << endl;
        }else{
            cout << "The books of category '" << CategoryName << "' : " << endl;
            // p->displayBooks();
            Book* pp=p->getBooks();
            while (pp!=nullptr){
                cout << "----->Book name : " << pp->getbookName() << endl;
                cout << "\tAuthor name : " << pp->getauthorName() << endl;
                pp=pp->getNextBook();
            }

        }
    };
```
5)
```
int Total(){
        int total=0;
        Category* p=Categories;
        while(p!=nullptr){
            total+=p->getNumberOfBooks();
            p=p->getNextCategory();
        }
        return total;
    };
```
6)
```
void deleteCategory(string cn){
        if(Categories->getCategoryName()==cn){
            Category* p=Categories;
            Categories=p->getNextCategory();
            delete p;
        }else{
            Category* p=Categories;
            while(p!=nullptr && p->getCategoryName()!=cn){p=p->getNextCategory();}
            if(p!=nullptr){
                Category* k=Categories;
                while((k->getNextCategory())->getCategoryName()!=cn){
                    k=k->getNextCategory();
                }
                k->setNextCategory(p->getNextCategory());
                delete  p;
            }
        }
    }
```

with the destructors :

```
~Category(){
        Book* a[NumOfBooks];
        int i=0;
        Book* p=books;
        while (p!=nullptr){
            a[i++]=p;
            p=p->getNextBook();
        }
        books=nullptr;
        for(int j=0;j<NumOfBooks;j++){
            delete a[j];
        }
    };
~Book(){
      delete nextBook;
  };
```