Nom: .....................................Prénom: ................................ الإسم: ........................ اللقب: ...................... الفوج: .......

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 **Question 1**: Cocher les énoncés corrects (note: 0 énoncé non coché, +1 énoncé correct coché, -1 énoncé incorrect coché) **(.........../5points) (A)**

Signaler les erreurs dans les codes suivants :

**class** C{

 **int** x; **int** y;

};

**void** **main**(){

 C o=C();

 o.x=3;

 x=4;

 cout<<"o.y";

}

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

**Exercice 1: (TAD) (.........../5points)**

1. On veut rédiger une spécification algébrique pour les files d'éléments. On définit les opérateurs suivants: **File\_Vide** (créer une file vide), **Enfiler** (ajouter un élément à la fin de la file: qui sera une queue de la file), **Défiler** (supprimer l'élément à la tête de la file), **Test\_Vide** (retourne vrai si la file est vide), **Tete** (consulter l'élément qui existe dans la tête de la file), **Queue** (consulter l'élément qui existe dans la fin de la file) . Proposez une spécification algébrique complète de ce TAD. Remarque Importante (pour trouver les axiomes, on considère uniquement les opérations File\_Vide et Enfiler comme constructeurs auxquelles on compose les 4 autres opérateurs)

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

**Genre** File

**Sortes** : Elément, Booléen

**Syntaxe :**

File\_Vide: 🡪 File

Enfiler: File x Elément🡪File

Défiler : File 🡪File

Test\_Vide : File 🡪Booléen

Tete : File🡪Elément

Queue : File🡪Elément

**Sémantique** :

**Précondition** :

Défiler(F) déf\_ssi not Test\_Vide(F)

Tete(F) déf\_ssi not Test\_Vide(F)

Queue(F) déf\_ssi not Test\_Vide(F)

**Axiomes : e :Elément, F, F’ :File**

Défiler(Enfiler(F,e)) == Enfiler(Défiler(F),e)

Test\_Vide(Enfiler(F,e)) ==Faux

Test\_Vide(File\_Vide()) ==Vrai

Tête(Enfiler(F,e)) ==Tête(F)

Queue(Enfiler(F,e))==e

**Fin\_Genre ;**

2. On veut spécifier un TAD qui étend (donc une extension) du TAD File: **File\_ext**. On propose d'ajouter deux nouvelles opérations: **Vider** (un nouveau constructeur qui permet de vider une file en défilant tous ses éléments) et **Afficher** (un inspecteur qui permet d'afficher le contenu de la file en donnant en sortie un affichage dont la sorte est "**Ecran**". On supposer l'existence d'un opérateur **Afficher\_élément** défini pour afficher des éléments de la sorte **Elément** sur l'écran. Dans la sorte **Ecran,** l’opérateur « + » permet d’afficher plusieurs élément (exemple : **afficher(e1)+afficher(e2)** permet d’afficher les éléments sur écran dans cet ordre **e1e2**).

**Afficher** consiste à afficher les éléments de la file depuis sa queue jusqu'à sa tête et elle peut utiliser l'opérateur **Afficher\_élément** défini pour afficher des éléments de la sorte **Elément** sur l'écran). Proposer une spécification de **File\_ext** (attention à la mise jour de la partie axiomatique). (**5points**)

**Genre** File\_ext **étend** File

**Sortes** : Elément, Booléen, Ecran

**Syntaxe :**

Vider: File 🡪 File

Afficher: File 🡪Ecran

**Sémantique** :

**Pré-condition** :

Afficher(F) **déf\_ssi** not Test\_Vide(F)

**Axiomes : e :Elément, F, F’ :File**

Afficher(Enfiler(F,e)) == Afficher\_élément(e)+ Afficher(F)

**Fin\_Genre ;**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

**Exercice 2: (POO) (............/5points)**

Proposer une classe implémentant la spécification algébrique précédente (File d'entier) en langage C++. On veut une implémentation en utilisant les tableaux.

**#include** <iostream>

**using** **namespace** std;

**const** **int** Max=20;

**class** File{

 **int** tete;

 **int** queue;

 **int** max;// doit etre inférieur à max

 **int** T[Max];

**public**:

 File (**int** maximum); // Créer une file vide

 **bool** **fileVide** (); //tester si la file est vide

 **void** **enfiler** (**int** valeur); // enfiler une valeur

 **void** **defiler** (); // défiler à une valeur

 **int** **get\_queue**();

 **int** **get\_tete**();

};

File::**File**(**int** maximum){

 max=maximum;

 tete=(-1);

 queue=(-1);

}

**bool** File::**fileVide**(){

 **return** queue==(-1);

}

**void** File::**enfiler**(**int** valeur){

 **if** (queue<max-1) {

 **if** (tete==(-1)){

 tete++;queue++;

 T[tete]=valeur;

 }

 **else**{

 queue++;

 T[queue]=valeur;

 }

 //cout<<sommet<<endl;

 //cout<<"T["<<sommet<<"]="<<T[sommet]<<endl;

 }

 **else**{

 cout<<"File Pleine"<<endl;

 }

}

**void** File::**defiler** (){

 **if** (**not** fileVide()){

 queue--;

 }

 **else** cout<<"file est vide"<<endl;

 // dépiler à une valeur

}

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….

………………………………………………………………………………………………………………………………………………………………………………………………………………………………….