NOM :

PRÉNOM :

PLACE :

Faculté d'Orsay, Licence 1, Semestre 1, Programmation Impérative **Année 2023-2024**

Examen mi-semestre du vendredi 27 octobre 2023 (une heure trente)

Calculatrices, téléphones mobiles et tout appareil électronique non autorisé doivent être éteints et déposés avec vos affaires personnelles. Seul document autorisé : une feuille au format A4 avec, au recto, le résumé de la syntaxe C++ de la fiche de TD 2 et, au verso, des notes manuscrites. Pour les étudiants inscrits en Français Langue Étrangère, un dictionnaire est autorisé.
Les exercices sont indépendants les uns des autres ; il n'est pas nécessaire de les faire dans l'ordre ; ceux marqués d'un ♣ sont plus difficiles mais font partie du barème sur 100. Le barème est indicatif et sujet à ajustements.
Les réponses sont à donner, sur le sujet ; en cas de besoin, utiliser la dernière page et mettre un renvoi.

Exercice 1 (Cours (6 points))**.**
Rappeler la syntaxe et la sémantique de la boucle `do ... while` en C++.

Exercice 2 (Boucles for : (7 points))**.**
Complétez la fonction suivante, en soulignant le compteur et entourant l'accumulateur :

```
/** Renvoie la somme des diviseurs de n
 * @param n: un entier
 * @return: un entier: la somme des diviseurs de n
 **/
int sommeDiviseurs(int n) {




}
```

Exercice 3 (Fonctions et conditionnelles filées (12 points))**.**
Pour postuler à un semestre aux États-Unis, vous devez traduire toutes vos notes dans le système américain en vous basant sur le tableau de correspondance suivant.

Note française	Note américaine
Entre 13 et 20	A
Entre 10 et 13 (exclu)	B
Entre 7 et 10 (exclu)	C
Entre 0 et 7 (exclu)	F

Pour cela vous allez implanter une fonction `noteAméricaine` qui prend en entrée votre note sur 20 et renvoie la lettre correspondante. Si la note n'est pas entre 0 et 20, la fonction doit renvoyer « `?` » (nous verrons plus tard comment signaler proprement une erreur) :

1. Écrivez des tests pour les notes 10 et $7,5$:

2. Implantez la fonction :

3. Donnez un exemple d'utilisation de cette fonction déclarant une nouvelle variable `info111` et y affectant la note américaine correspondant à 18.

Exercice 4 (Tableaux 1D et fonctions (20 points))**.**

1. Implantez une fonction `initialise` prenant un entier `taille` en paramètre et renvoyant un tableau de booléens de la taille donnée, dont les deux premières valeurs sont `false` et les suivantes sont `true` :

2. Complétez la fonction suivante qui met à `false` toutes les valeurs du tableau dont l'indice est un multiple de `d` autres que `d` lui-même :

```
vector<bool> élimineMultiples(vector<bool> t, int d) {




    return t;
}
```

3. Complétez la fonction suivante qui renvoie tous les entiers `d` tels que `t[d]` est vrai :

```
vector<int> sélectionne(vector<bool> t) {




    return résultat;
}
```

4. ♣ Complétez le test de la fonction `mystère` donnée ci-dessous :

```
vector<bool> mystère(int taille) {
    vector<bool> résultat = initialise(taille);
    for ( int d = 2; d < taille; d++ )
        if ( résultat[d] )
            résultat = élimineMultiples(résultat, d);
    return résultat;
}
```

```
CHECK( sélectionne(mystère(15)) ==

```

Que fait cette fonction ?

Exercice 5 (Pile, tas et exécution pas à pas (15 points))**.**
On considère le fragment de programme suivant :

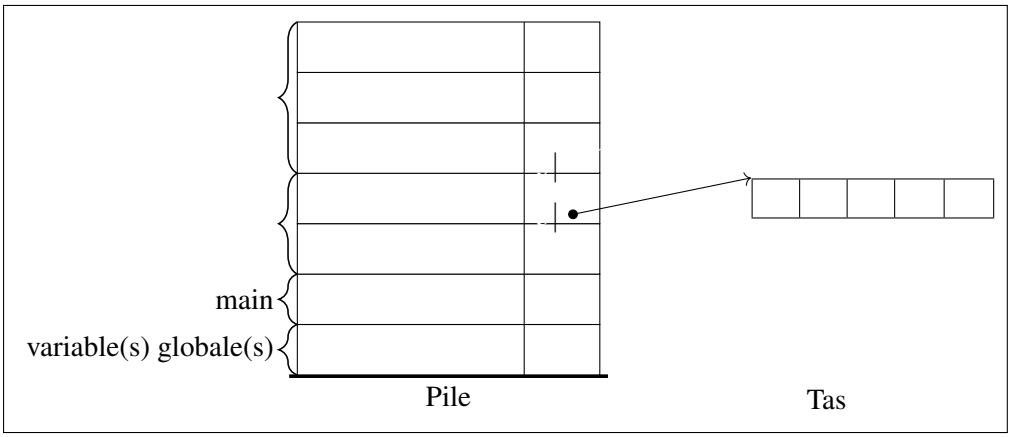
```
int a;

void g(vector<int> t, int i) {
    int k = a + i;
    t[i] = k;
    // ICI
}

void f(int b) {
    a = a - 1;
    b = b + 1;
    vector<int> t;
    t = vector<int>(b);
    for ( int i = 0; i < 3; i++ ) {
        t[i] = 2 * i;
    }
    g(t, b - 1);
}

int main() {
    a = 4;
    int b = a;
    f(b);
    // LÀ
    cout << a << " " << b << endl;
    return 0;
}
```

1. Soulignez la ou les déclaration(s) de paramètre(s) formel(s).
2. Encadrez d'un rectangle la ou les déclaration(s) de variable(s) locale(s).
3. Entourez d'un rond la ou les déclaration(s) de variable(s) globale(s).
4. Sur votre brouillon, exécutez pas-à-pas le programme. En suivant les conventions du cours, complétez ci-dessous le dessin de la pile et du tas au moment où l'exécution atteint la ligne marquée « `ICI` ».



5. Quelles sont les valeurs des variables suivantes au moment où l'exécution du programme atteint la ligne marquée « `LÀ` » ?
   — variable globale `a` :
   — variable locale `b` :

Exercice 6 (Mastermind (40 points))**.**
Le MASTERMIND est un jeu de société où un joueur doit deviner un code secret choisi par un arbitre. Le code est composé de quatre jetons de couleurs parmi six couleurs possibles, placés dans un ordre donné. La même couleur peut être utilisée plusieurs fois. Le joueur doit deviner le code en faisant des propositions. À chaque proposition, l'arbitre donne les informations suivantes :

— Le nombre de pions de la bonne couleur et à la bonne position ;

— Le nombre de pions de la bonne couleur mais en mauvaise position.

Ces nombres sont donnés sur le plateau de jeu par autant de fiches rouges et blanches, respectivement. La manche se termine lorsque le joueur a découvert le code, ou lorsque le nombre maximal de propositions est atteint (douze en tout sur le plateau de la figure 1). L'arbitre et le joueur inversent alors leurs rôles.

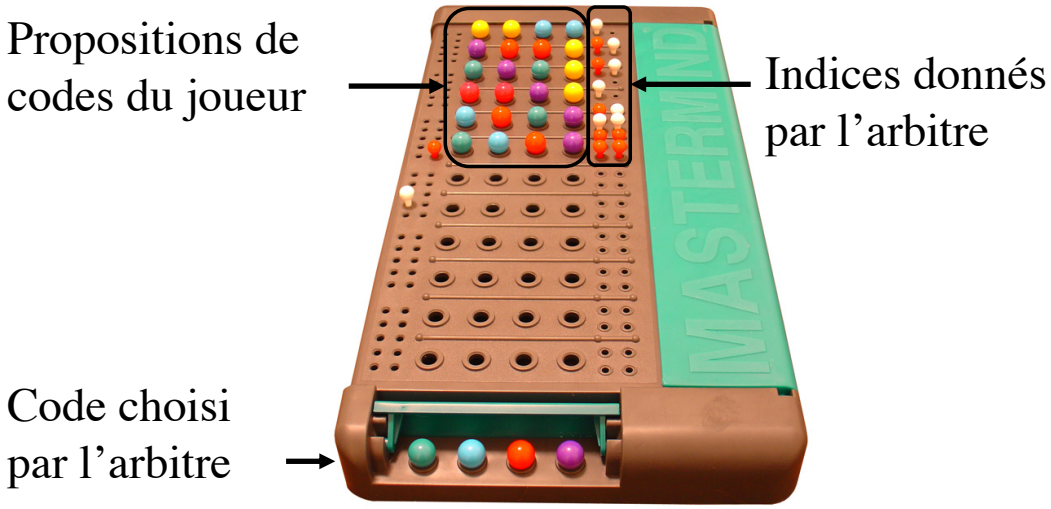


FIGURE 1 – Plateau de jeu de Master Mind

On se propose d'implanter MASTERMIND en automatisant l'arbitre qui choisit le code et donne les indices. On encodera les six couleurs par des entiers allant de 0 à 5.

1. On s'intéresse dans un premier temps aux éléments du code qui sont à la bonne position. Implantez une fonction `nombrePositionsCorrectes` qui prend en paramètres deux tableaux d'entiers de même taille et renvoie le nombre de cases identiques, c'est à dire avec la même valeur à la même position.

2. On s'intéresse à présent aux éléments de la proposition du joueur qui sont présents dans le code secret de l'arbitre mais pas forcément à la bonne position. Pour cela, on se servira de la fonction mystère suivante :

```
vector<int> mystère(vector<int> code) {
    vector<int> res = {0, 0, 0, 0, 0, 0};
    for ( int i = 0; i < code.size(); i++ ) {
        int indice = code[i];
        res[indice] = res[indice] + 1;
    }
    return res;
}
```

(a) Complétez les tests suivants et ajoutez un quatrième test :

```
CHECK( mystère( {1, 1, 1, 1} ) ==
       vector<int>({0, 4, 0, 0, 0, 0}) );
CHECK( mystère( {5, 5, 5, 5} ) ==

CHECK( mystère( {0, 1, 2, 3} ) ==
```

(b) Proposez une documentation pour cette fonction qui décrit le plus précisément possible comment elle fonctionne.

(c) Écrivez une fonction `nombreValeursCorrectes` qui prend en paramètres deux tableaux d'entiers et renvoie le nombre d'éléments présents dans les deux tableaux, correctement positionnés ou pas. La fonction `nombreValeursCorrectes` doit vérifier les tests suivants :

```
CHECK( nombreValeursCorrectes({0, 0, 1, 1}, {2, 2, 3, 3}) == 0 );
CHECK( nombreValeursCorrectes({0, 0, 1, 1}, {0, 0, 1, 1}) == 4 );
CHECK( nombreValeursCorrectes({0, 0, 1, 1}, {2, 1, 2, 2}) == 1 );
```

Indice : On pourra utiliser par exemple la fonction `mystère` et une fonction `min` que l'on considère déjà définie et qui renvoie le minimum entre deux entiers.

3. Écrivez maintenant une instruction pour calculer le nombre de valeurs correctes mal positionnées, et stocker le résultat dans une variable `malPlaces` :

4. ♣ Décrire informellement comment automatiser les propositions du joueur pour trouver le code en cherchant à réduire le nombre de propositions.
   Indice : Considérez l'ensemble des propositions possibles et comment réduire cet ensemble au fur et à mesure des indices de l'arbitre.