L2- Info4A **Examen Session 2** 2023-2024

## Exercice (5 points)

Considérons une classe `BankAccount` qui représente un compte bancaire. Le compte bancaire est caractérisé par les attributs suivants :

- `accountNumber` (numéro de compte)
- `accountHolder` (titulaire du compte)
- `balance` (solde)

La classe `BankAccount` doit inclure les fonctionnalités suivantes :

- Un constructeur qui initialise le numéro de compte, le titulaire du compte et le solde initial (par défaut, le solde initial est de 0).
- Des fonctions d'accès (`getters`) pour le numéro de compte, le titulaire du compte et le solde.
- Une fonction `deposit` pour effectuer un dépôt sur le compte. Cette fonction prend un montant en paramètre et met à jour le solde du compte en ajoutant ce montant.
- Une fonction `withdraw` pour effectuer un retrait du compte. Cette fonction prend un montant en paramètre et met à jour le solde du compte en le déduisant, à condition que le solde soit suffisant. Sinon, un message d'erreur approprié doit être affiché.
- Une fonction `display` qui affiche les détails du compte (numéro de compte, titulaire du compte et solde).

**Questions :**

1. Déclarez la classe `BankAccount` avec les attributs et les fonctions mentionnés ci-dessus.
2. Implémentez le constructeur de la classe `BankAccount`.
3. Implémentez les fonctions d'accès (`getters`) pour les attributs de la classe `BankAccount`.
4. Implémentez la fonction `deposit` et la fonction `withdraw` en tenant compte de la vérification du solde.
5. Implémenter la fonction `display`.
6. Proposer des tests dans la fonction `main`.

# Examen info4A - année 2022/2023 - session 2

## Partie langage C

Vous devez écrire vos réponses directement sur le sujet, aux emplacements grisés situés immédiatement après les questions. Votre nom doit être écrit sur la partie cachée de la copie double. Votre numéro d'anonymat officiel doit être écrit sur la copie double et sur le sujet qui devra être inséré dans la copie double.

Durée indicative : 1h30.

Documents autorisés (conjointement avec la partie C++) : 4 feuilles A4 recto-verso, avec contenu libre imprimé et/ou manuscrit.

Calculatrices et ordinateurs interdits.

## Partie A (2 points)

### Question 1 (2 points)

On rappelle que, par définition, pour tout entier naturel $x > 0$, $x! = \Pi_{i=1}^{x} i$.

Définissez la fonction...

```
int ratioFact(int n, int k);
```

...qui, en supposant que $k > 0$, $n > 0$ et $k \leq n$, calcule et retourne la valeur $n!/k!$ (factorielle $n$ sur factorielle $k$). Vous devez réaliser un code simple et efficace en minimisant le nombre de multiplications.

```
int ratioFact(int n, int k)
{









}
```

## Partie B (3 points)

### Question 2 (1.5 point)

Donnez les valeurs affichées par l'exécution des lignes de code suivantes :

```
printf("%d\n", 0b00001111 << 0b11);
printf("%x\n", (7 | 8));
printf("%x\n", (0xF0 & 0x71) | 0x2);
```

## Question 3 (1.5 point)

Sans utiliser de boucle, définissez la fonction...

```
uint16_t f(uint16_t m, int k);
```

...qui produit le mot binaire obtenu en complémentant les $k$ derniers bits du mot binaire représenté par le paramètre $m$.

Par exemple, si $m$ est le mot `0000111100001111` et $k$ vaut 5, alors la valeur de retour doit représenter le mot :

```
0000111100010000
```

(Il existe plusieurs solutions, dont une utilisant notamment l'opérateur bitwise ou exclusif `^`).

```
uint16_t f(uint16_t m, int k)
{









}
```

## Partie C et E (5 points)

Soit `VMAX` une constante globale. Dans la suite, on considérera que cette constante a la valeur 7, mais les réponses aux questions devront rester correctes si on change cette valeur.

```
#define VMAX 7
```

Dans la suite, on représente un ensemble d'entiers compris entre 0 et `VMAX` par un tableau de `VMAX+1` cellules contenant chacune un Booléen représenté par un entier pouvant avoir pour valeur 0 ou 1.

Voici quelques exemples d'ensembles et leurs représentations avec la convention utilisée :

| Ensemble | Représentation par un tableau d'entiers |
|---|---|
| $\{\}$ | `0,0,0,0,0,0,0,0` |
| $\{0,1,2,3\}$ | `1,1,1,1,0,0,0,0` |
| $\{0,1,2,3,4,5,6,7\}$ | `1,1,1,1,1,1,1,1` |

## Question 4 (1 point)

Définissez la fonction...

```
void u(int e1[], int e2[], int r[]);
```

...qui place dans le tableau désigné par `r` la représentation de l'union des ensembles représentés par les tableaux désignés par les paramètres `e1` et `e2`. Les trois tableaux sont supposés être de taille `VMAX+1`.

```
void u(const int e1[], const int e2[], int r[])
{




}
```

## Question 5 (2 points)

Complétez les lignes de codes ci-dessous...

```
int a[VMAX+1] = {0,1,1,0,0,1,0,0};
int b[VMAX+1] = {1,1,0,0,0,1,1,0};
int* c =
```

...de manière à ce que à l'issue de leur exécution, la variable `c` pointe une tableau situé dans le tas représentant l'union des ensembles représentés par les tableaux `a` et `b`. Vous devez utiliser la fonction `u` de la question précédente (en supposant qu'elle est correctement implémentée).

## Question 6 (2 points)

Dessinez la représentation détaillée de toutes les données en mémoire, dans la pile et dans le tas, à l'issue de l'exécution des lignes de code de la question précédente (en supposant qu'elles aient été correctement programmées).

## Parties D et F (5 points)

On représente une liste de chaînes de caractères avec la structure suivante :

```
typedef struct
{
    char** tab;
    int n;
    int capa;
}tokens;
```

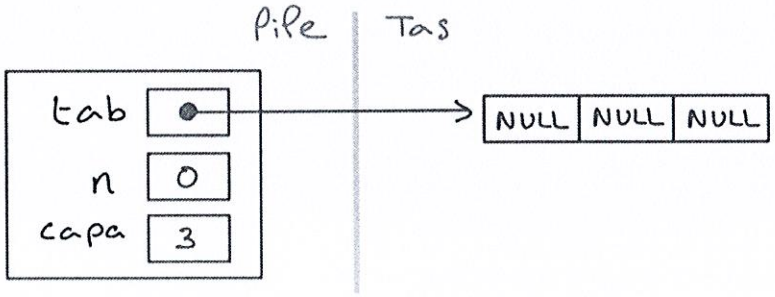
Le champ `n` indique le nombre de chaînes stockées dans la liste.

Le champ `tab` désigne un tableau de pointeurs sur `char` dont chaque cellule d'indice inférieur à `n` contient un pointeur désignant une chaîne située dans le tas, et chaque cellule d'indice au moins égal à `n` contient la valeur `NULL`.

Le champ `capa` indique la nombre maximum de chaînes pouvant être stockées, ce qui correspond à la taille du tableau pointé par `tab`.

### Question 7 (1 point)

Donnez les lignes de code permettant d'obtenir la situation suivante en mémoire.



## Question 8 (1 point)

Définissez la fonction...

```
void affTokens(tokens* w);
```

... qui affiche les chaînes contenues dans la liste désignée par `w`.

```
void affTokens(tokens* w)
{









}
```

## Question 9 (1 point)

Définissez la fonction...

```
void concToken(tokens* w, int i, char* dest);
```

...dont l'exécution a l'effet suivant :

- Aucun effet si le nombre de chaînes stockées dans la liste désignée par `w` est plus petit que `i`.
- Sinon, la chaîne située en position `i` dans la liste désignée par `w` est concaténée à la chaîne désignée par `dest`.

On suppose que `dest` pointe un tableau de caractères contenant une chaîne (possiblement vide). On suppose que la taille de ce tableau est suffisante pour contenir le résultat de la concaténation.

```
void concToken(tokens* w, int i, char* dest)
{









}
```

## Question 10 (2 points)

Définissez la fonction...

```
void videListe(tokens* w);
```

...dont l'exécution a pour effet de vider la liste désignée par `w`, c'est à dire de retirer toutes les chaînes qui s'y trouvent et de libérer la mémoire occupée par ces chaînes.

```
void videListe(tokens* w)
{









}
```