Nom: **Prénom:** **Numero:**

Partiel de ProgASD

23 mars 2022

Durée: 2h. Le seul document autorisé est une feuille au format A4 sur laquelle vous pouvez avoir consigné des notes manuscrites personnelles. Tous les exercices sont indépendants. Si vous ne savez pas répondre à une question, vous pouvez tout de même utiliser sa réponse pour les questions suivantes.

Une grande importance sera accordée à la qualité de la rédaction (lisibilité, indentation, ...).

1 – Questions de cours (5pt)

Pour chacune des questions suivantes, répondez en une ou deux phrases courtes et précises.

1. *(0,5pt)* Les variables globales sont-elles allouées de manière statique ou dynamique? Justifiez.

2. *(0,5pt)* Où se termine la portée d'une variable locale dans une fonction ?

3. *(1,0pt)* On suppose que `p` est un entier, comment doit être déclarée la variable `q` afin que l'expression `q = &p;` soit correcte ? Quelle est sa signification ?

4. *(1,0pt)* Donnez la ligne de code permettant d'allouer un bloc de mémoire dynamique permettant de stocker 3 entiers.

5. *(0,5pt)* Quelle est l'utilité du pointeur `NULL` ?

6. *(0,5pt)* Que contient une variable avant son initialisation ?

7. *(1,0pt)* Comment lors de leur déclaration, le compilateur fait-il la différence entre une variable locale, une variable globale et un paramètre de fonction ?

2 – Sémantique (5pt)

On considère le programme suivant, le calcul qu'il réalise n'a pas d'importance, nous nous intéressons ici seulement à son implémentation.

```
1  int divmod(int a, int b, int &m) {
2      m = a % b;
3      return a / b;
4  }
5  int special(int n) {
6      int i, s = 1;
7      for (i = 1; i < n; i++) {
8          int d, m;
9          d = divmod(i, 3, m);
10         s = s * (d + m);
11     }
12     return s;
13 }
14 int main() {
15     int x = special(10);
16     cout << x;
17     return 0;
18 }
```

Pour répondre aux questions suivantes, utilisez les tableaux page 7.

1. *(1pt)* Représentez les tableaux d'activation des trois fonctions `divmod`, `special` et `main`.

2. *(4pt)* En utilisant les conventions vues en cours et en TD, représentez l'état de la mémoire aux 4 étapes suivantes. À chaque étape, vous indiquerez le nom des variables associées à chaque case, lorsque celles-ci sont accessibles depuis le programme :

(a) Dans `special` avant l'exécution de la ligne 7.
(b) Lors du premier appel à `divmod` avant l'exécution de la ligne 3.
(c) Lors du deuxième appel à `divmod` avant l'exécution de la ligne 2.
(d) Dans `special` après l'exécution de la ligne 10 au troisième tour de boucle.

Nom: **Prénom:** **Numero:**

3 – Pointeurs (4pt)

Attention: *Dans cet exercice il est vous est demandé de ne pas utiliser de références mais des pointeurs !*

Soit les deux structures suivantes permettant de représenter un point en deux dimensions ainsi qu'un triangle à partir de ses trois sommets :

```
1 struct point {
2     float x, y;
3 };
4 struct triangle {
5     point a, b, c;
6 };
```

1. *(1pt)* Écrivez la fonction `distance` prenant en paramètres deux *pointeurs vers des points* et renvoyant un réel, qui calcule la distance entre deux points à l'aide de la formule suivante :

$$|ab| = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

2. *(1pt)* Écrivez la fonction `milieu` qui prend en paramètres deux *pointeurs vers des points* et renvoyant un point, qui calcule le milieu du segment défini par les deux points à l'aide de la formule suivante :

$$M = \left(\frac{x_a + x_b}{2}, \frac{y_a + y_b}{2} \right)$$

3. *(2pt)* Écrire la fonction `circonscrit` qui vérifie si un triangle donné en paramètre est rectangle et transmet le centre de son cercle circonscrit.

- le triangle devra être passé en paramètre sous la forme d'un pointeur;
- la fonction doit renvoyer un booléen indiquant si le triangle est rectangle, pour cela vérifier qu'il satisfait le théorème de Pythagore;
- si le triangle est rectangle, le centre du cercle circonscrit est le milieu de l'hypothénuse (son plus grand côté) et doit être transmis via un paramètre résultat;
- si le triangle n'est pas rectangle, la fonction doit renvoyer faux et la valeur du paramètre résultat n'est pas spécifiée.

4 – Structures et références (6pt)

On considère un système de gestion de stock pour des entrepôts. Soit la structure suivante permettant de définir un `produit` par son nom, son prix et son volume unitaire ainsi que la quantité de ce produit disponible ; et la structure `entrepot` qui définit la capacité totale de stockage d'un entrepôt ainsi que la liste des produits qui y sont stockés.

```
1  struct produit {
2      string nom;
3      int prix;
4      int quantité;
5      int volume;
6  };
7  struct entrepot {
8      int capacite;
9      vector<produit> stock;
10 };
```

Nom: **Prénom:** **Numero:**

1. *(1pt)* Écrire la fonction `lireproduit` qui permet de saisir au clavier les informations relatives à un produit et de le transmettre en paramètre résultat.

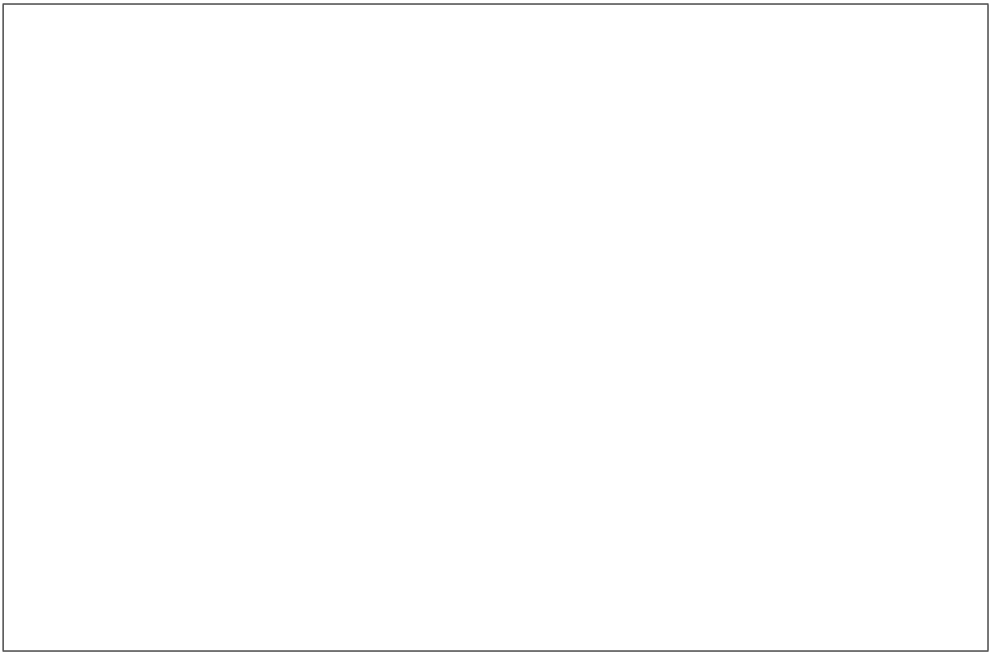
2. *(1,5pt)* Écrire la fonction `achat` qui prend en paramètre un produit et la quantité désirée de ce produit. Cette fonction doit mettre à jour le produit, transmettre en résultat le prix à facturer au client et renvoyer la quantité de produit qu'il va réellement recevoir (en fonction du stock disponible).

3. *(1,5pt)* Écrire la fonction `espace_disponible` qui prend en paramètre un entrepot et renvoie le volume de stockage restant disponible.

4. *(2pt)* Écrire la fonction `commande`, dont le prototype vous est donné, qui réalise une commande pour un client. Cette fonction prend en paramètre un entrepot où les produits doivent être retirés et une commande qui indique pour chaque produit

la quantité désirée par le client. Elle renvoie directement le prix total à facturer au client et dans le paramètre résultat `manque` le nombre de produits manquants dans le stock qui ne seront donc pas livrés au client.

```
int commande(entrepot &e, vector<int> &commande, int &manque)
```



Nom: **Prénom:** **Numero:**

	15	
	14	
	13	
	12	
	11	
	10	
	9	
	8	
	7	
	6	
	5	
	4	
	3	
	2	
	1	
	0	

	15	
	14	
	13	
	12	
	11	
	10	
	9	
	8	
	7	
	6	
	5	
	4	
	3	
	2	
	1	
	0	

	15	
	14	
	13	
	12	
	11	
	10	
	9	
	8	
	7	
	6	
	5	
	4	
	3	
	2	
	1	
	0	

	15	
	14	
	13	
	12	
	11	
	10	
	9	
	8	
	7	
	6	
	5	
	4	
	3	
	2	
	1	
	0	

Structures de contrôle :

```
#include <iostream>
using namespace std;
void main() {
    ...
}
```

```
cin >> n;
cout << 3 * n + 1 << endl;
```

```
if (x == 1) {
    ...
} else {
    ...
}
```

```
for (int i = 0; i < 10; i++) {
    ...
}
```

```
while (i <= 10) {
    ...;
}
```

```
do {
    ...
} while (i <= 10);
```

```
int i;
...
switch (i) {
    case 1: res = "un"; break;
    case 2: res = "deux"; break;
    ...
    default: res = "?"; break;
}
```

Fonctions, références et pointeurs :

```
/** La fonction toto
 * n une donnée
 * m un résultat
 * r une donnée/résultat
 * retourne un entier
 **/
int toto(int n, int &m, int &r){
    int res;
    ...
    return res;
}
```

```
int a = 3;
int &b = a;
b = 5;
```

```
int a = 3;
int *b;
b = NULL;
b = &a;
*b = 5;
```

Types de données :

```
struct personne {
    string nom, prenom;
    int naissance;
};
...
personne p;
p.naissance = 1933;
cout << p.naissance;
```

```
enum jours { lundi, mardi, ...
  dimanche };
```

```
vector<int> v;
...
v[5] = 3;
v.push_back = 7;
```