C++ - Examen final du 28 Janvier 2000
Correction

1 - Rappels concernant les modalités d'examen[1]

Il s'agit d'un examen individuel, et les consignes les plus strictes ont été données au personnel assurant la surveillance de l'épreuve, de façon à garantir le respect de cette règle.

La consultation de documents écrits est autorisée, dans la mesure où elle s'effectue sans rendre le caractère individuel de l'examen incontrôlable.

Il s'agit d'une épreuve écrite. Lorsqu'une indication concernant la longueur de la réponse attendue figure dans l'énoncé, le respect de cette consigne sera pris en compte lors de la notation.

La durée maximale de composition est fixée par le règlement d'étude à deux heures. Les candidats n'ayant pas remis leur copie à l'expiration de ce délai seront considérés comme ayant rendu copie blanche.

2 - Savez-vous lire le C++ ?

Les dix question suivantes comportent un fragment de code et plusieurs affirmations le concernant. Vous devez, pour chacune de ces affirmations, indiquer si elle est Vraie ou Fausse. Vos réponses devraient donc apparaître sous une forme analogue à ceci :

`x = y + 3; //Exemple de question`		
Cette ligne place dans la variable `x` le résultat obtenu en ajoutant 3 au contenu de la variable `y`.	V	
Cette ligne nécessite que les variables `x` et `y` aient été préalablement définies.	V	
Cette ligne causera une erreur d'exécution si la valeur contenue dans la variable `x` n'est pas égale au résultat obtenu en ajoutant 3 au contenu de la variable `y`.		F

Comme le montre cet exemple, il est possible que plusieurs des affirmations proposées soient vraies. Il est également possible qu'elles soient toutes fausses.

`int a(12); //Question a`		
Cette ligne appelle la fonction `a()` en lui passant comme paramètre la valeur entière 12.		F
Cette ligne déclare une fonction `a()` qui accepte comme paramètre un entier, dont la valeur par défaut est 12.		F
Cette ligne définit une variable entière `a`, dont la valeur maximale admissible est 12.		F
Cette ligne définit une variable entière `a`, initialisée avec la valeur 12.	V	

`int b(int i = 12); //Question b`		
Cette ligne appelle la fonction `b()` en lui passant comme paramètre la valeur entière 12.		F
Cette ligne déclare une fonction `b()` qui accepte comme paramètre un entier, dont la valeur par défaut est 12.	V	
Cette ligne causera une erreur lors de la compilation du programme.		F
Cette ligne définit une variable entière `b` initialisée avec la valeur 12.		F
Cette ligne nécessite que la variable `i` ait été préalablement définie.		F

[1] Ces modalités figurent explicitement dans la Leçon 0, et sont identiques à celles du partiel du 26 Novembre. Nul n'est donc sensé les découvrir le jour de l'examen.

`c(12); //Question c`		
Cette ligne appelle la fonction `c()` en lui passant comme paramètre la valeur entière 12.	V	
Cette ligne déclare une fonction `c()` qui accepte comme paramètre un entier, dont la valeur par défaut est 12.		F
Cette ligne causera certainement une erreur lors de la compilation du programme.		F
Cette ligne appelle 12 fois la fonction `c()`.		F
Cette ligne nécessite que la variable `c` ait été préalablement définie.		F

`d(); //Question d`		
Cette ligne appelle la fonction `d()`.	V	
Cette ligne nécessite que la fonction `d()` ait été préalablement déclarée.	V	
Cette ligne déclare une fonction `d()` qui ne retourne aucun résultat et n'accepte aucun paramètre.		F
Pour que cette ligne puisse être acceptable, il est nécessaire que la fonction `d()` soit dépourvue de paramètre.		F

`e(j); //Question e`		
Cette ligne appelle la fonction `e()`.	V	
Cette ligne nécessite que la variable `j` ait été préalablement définie.	V	
Pour que cette ligne puisse être acceptable, il est nécessaire que la fonction `e()`accepte un paramètre d'un type compatible avec celui de la variable `j`.	V	

Si la fonction `e()` est définie ainsi
`void e(int k)`
`{`
`k = k * 2 ;`
`}`
le contenu de la variable passée comme paramètre sera modifié par l'exécution de cette ligne.
		F

Si la fonction `e()` est définie ainsi
`void e(int j)`
`{`
`j = j * 2 ;`
`}`
le contenu de la variable passée comme paramètre sera modifié par l'exécution de cette ligne.
		F

`int * f; //Question f`		
A la suite de cette ligne, on peut écrire `f = 4;`		F
A la suite de cette ligne, on peut écrire `* f = 4;`		F
A la suite de cette ligne, on peut écrire `& f = 4;`		F
A la suite de cette ligne, on peut écrire `f = new int;`	V	

`int m; //Question g`
`int * g = & m;`		

A la suite de ces lignes, on peut écrire `g = 4;`		F
A la suite de ces lignes, on peut écrire `* g = 4;`	V	
A la suite de ces lignes, on peut écrire `& g = 4;`		F
A la suite de cette ligne, on peut écrire `g = new int;`	V	

`int h[6]; //Question h`		
A la suite de cette ligne, on peut écrire `h = 4;`		F
A la suite de cette ligne, on peut écrire `h[0] = 4;`	V	
A la suite de cette ligne, on peut écrire `h[6] = 4;`		F
A la suite de cette ligne, on peut écrire `h[] = {1, 2, 3, 4, 5, 6};`		F

`int *i = new int[6]; //Question i`		

A la suite de cette ligne, on peut écrire
`if (i != NULL)`
`i = 4;`
		F

A la suite de cette ligne, on peut écrire
`if (i != NULL)`
`i[0] = 4;`
	V	

A la suite de cette ligne, on peut écrire
`if (i != NULL)`
`i[6] = 4;`
		F

A la suite de cette ligne, on peut écrire
`if (i != NULL)`
`i[] = {1, 2, 3, 4, 5, 6};`
		F

`int j[6]; //Question j`
`traite(j,6);`		

Ces deux lignes ne posent aucun problème si la fonction `traite()` est déclarée ainsi `void traite(int j, int taille);`		F
Ces deux lignes ne posent aucun problème si la fonction `traite()`est déclarée ainsi `void traite(int *j, int taille);`	V	
Ces deux lignes ne posent aucun problème si la fonction `traite()`est déclarée ainsi `void traite(int *j, int taille = 10};`	V	
Ces deux lignes ne posent aucun problème si la fonction `traite()`est déclarée ainsi `int traite(int j);`		F

3 - Pouvez-vous écrire un peu de C++ ?

Attention, ces questions NE SONT PAS classées par ordre de difficulté croissante.

Question 1

Ecrivez une fonction nommée `nombreDeNegatifs()` capable de renvoyer le nombre de valeurs négatives contenues dans un tableau d'entiers qui lui est transmis comme paramètre.

```
int nombreDeNegatifs(int * tab, int taille)
{
int index;
int resultat = 0;
for (index = 0 ; index < taille ; index = index + 1)
  if(tab[index] < 0)
     resultat = resultat + 1;
return resultat;
}
```

Question 2

Ecrivez un fragment de code déclarant un tableau de dix entiers recevant (par initialisation ou affectation) des valeurs de votre choix, et qui appelle la fonction décrite à la Question 1 pour déterminer le nombre de valeurs inférieures à zéro contenues dans ce tableau.

```
int tableau[] = {-1,1,-2,3,-4,5,7,7,0,-11};
int nombre = nombreDeNegatifs(tableau, sizeof(tableau)/sizeof(tableau[0]));
```

Question 3

Ecrivez une fonction nommée `renverse()` capable d'inverser l'ordre des caractères d'un tableau qui lui est transmis comme paramètre.

En d'autres termes, après

```
char exemple[] = "BRAVO";
renverse(exemple);
```

la variable exemple doit contenir "OVARB".

```
#include "string.h"

void renverse(char * chaine)
{
int nbCaracteres = strlen(chaine);
char * temp = new char[nbCaracteres + 1];
strcpy(temp, chaine);
int index;
for(index = 0 ; index < nbCaracteres ; index = index + 1)
  chaine[index] = temp[nbCaracteres - index - 1];
delete temp;
}
```

Question 4

La classe Q4 est définie ainsi :

```
class Q4
{
public:
  int m_taille;
  double * m_adresse;
  bool creation(int tailleSouhaitee);
};
```

Comment peut-on définir la fonction membre `creation()` pour que son appel se traduise par la réservation d'une zone de mémoire permettant de stocker des valeurs décimales, sachant que :

- l'adresse de cette zone doit être stockée dans la variable `m_adresse`,
- le nombre de valeurs décimales qui doivent pouvoir être stockées dans la zone de mémoire est transmis comme paramètre à la fonction `creation()`
- `ce nombre de valeurs` doit être stocké dans la variable `m_taille`,
- la fonction `creation()` doit renvoyer la valeur `true` si l'allocation dynamique a été possible, et `false` si elle a échoué.

```
bool Q4::creation(int taille)
{
m_taille = taille;
m_adresse = new double[taille];
return (m_adresse != NULL);
}
```

Question 5

Ecrivez un fragment de code instanciant la classe Q4 et utilisant la fonction `creation()` pour rendre cette variable de type Q4 capable de stocker 100 valeurs.

```
Q4 superTableau;
superTableau.creation(100);
```