**TEST2 ICC MT & EL 23.11.2018 EPFL / R. Boulic**

**W=blanc, B=bleu, G=vert, Y=jaune, All = même réponse pour tous**

**1) (6 pts) Analyse de programmes**

**1.1) algorithme avec array**

```
1
2   #include <iostream>
3   #include <array>
4
5   using namespace std;
6
7   constexpr size_t DIM(10);
8
9   int main()
10  {
11      array<unsigned int,DIM> tab = {7,2,9,4};
12      int max_sum_successive_elem(0), sum(0);
13
14      for(size_t i(DIM-1) ; i >= 0 ; --i)
15      {
16          if(i-1 >= 0)
17          {
18              sum = tab[i] + tab[i-1];
19              if(sum > max_sum_successive_elem)
20                  max_sum_successive_elem = sum;
21          }
22      }
23      cout << max_sum_successive_elem << endl;
24      return 0;
25  }
26
```

Choisir parmi une des réponses proposées en justifiant votre réponse par deux à trois phrases (requis pour avoir les points) ; *selon la réponse choisie*, indiquer comment corriger le programme ou indiquer le résultat affiché :

**Ce programme ...**

- ➔ A : ne compile pas avec C++11 (standard utilisé dans le MOOC et les TPs)
- ➔ B : s'exécute jusqu'à la fin et affiche une réponse correcte
- ➔ C : s'exécute jusqu'à la fin et affiche une réponse imprévisible
- ➔ D: s'exécute mais s'arrête avant la fin avec un message du type « segmentation fault »

**Réponse choisie, avec sa JUSTIFICATION :**

**[All]** Le type **size_t** est un entier positif ou nul (MOOC p 44). Ligne 14, lorsque i vaut zéro, le motif binaire du résultat de l'évaluation de 0 – 1 est 1111…1111 dans la représentation du complément à 2. Ce motif binaire est interprété comme un nombre positif très grand qui produit une condition VRAI à la ligne 16. De ce fait la ligne 17 cherche à accéder à tab[i] avec cette valeur incorrecte d'indice et cela produit un segmentation fault.

**1.2) vector, fonction et lecture**

```
1
2   #include <iostream>
3   #include <vector>
4
5   using namespace std;
6
7   void fill_vector(vector<double>& v);
8   int  sum_class(const vector<double>& v);
9
10  int main()
11  {
12       const vector<double> test_case = {-0.25,1.,4.,3.25};
13       vector<double> vect;
14
15       fill_vector(vect);
16
17       if(sum_class(test_case) == 1)
18            if(vect.size() == test_case.size())
19                 cout << "same size and class"
20                     << endl;
21       return 0;
22  }
23
24  void fill_vector(vector<double>& v)
25  {
26       while(sum_class(v) <= 0)
27       {
28            double val(0.);
29            cin >> val;
30            v.push_back(val);
31       }
32  }
33
34  int sum_class(const vector<double>& v)
35  {
36       double sum(0);
37       if(v.size()  > 0)
38            for(size_t i(0); i < v.size(); ++i)
39                 sum+=v[i];
40
41       return (sum<0)?-1:((sum>0)?1:0);
42  }
```

Choisir parmi une des réponses proposées en justifiant votre réponse par quatre à cinq phrases (requis pour avoir les points) ; *selon la réponse choisie*, indiquer comment corriger le programme ou indiquer le résultat affiché si on frappe au clavier cet ensemble de valeurs avant de valider avec Enter :

```
-2.  1.  4.  3.      [W]
-2.  1.  2.  1.      [B]
-2.  2.  1.  1.      [G]
-2.  0.  3.  4.      [Y]
```

**W=blanc, B=bleu, G=vert, Y=jaune, All = même réponse pour tous**

**Ce programme ...**

- ➔ A : ne compile pas avec C++11 (standard utilisé dans le MOOC et les TPs)
- ➔ B : s'exécute jusqu'à la fin et affiche `same size and class`
- ➔ C : s'exécute jusqu'à la fin et n'affiche aucun message
- ➔ D: s'exécute mais s'arrête avant la fin avec un message du type « segmentation fault »

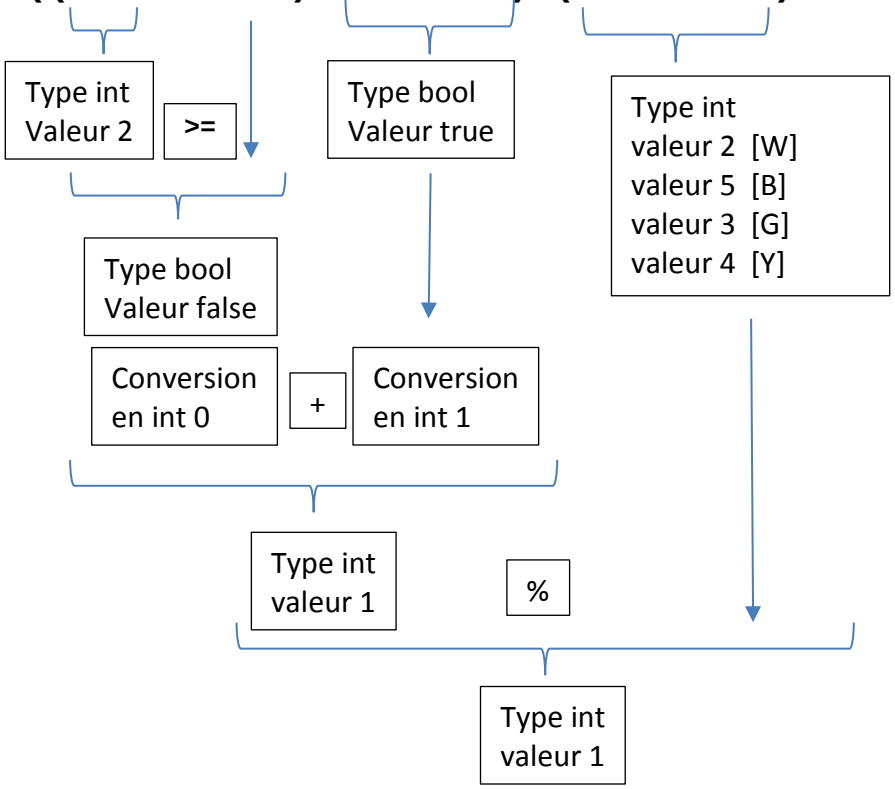
**Réponse choisie, avec sa JUSTIFICATION :**

[All] La fonction **fill_vector** modifie le vector passé par référence. Il est vide au moment de l'appel puis il est rempli élément par élément avec la méthode **push_back** (MOOC p 48) tant que la somme des éléments qu'il contient est négative ou nulle. Le calcul de la somme du vector est fait avec la fonction **sum_class** qui reçoit le vector par référence constante car le vector ne doit pas être modifié par cette fonction. Cette fonction renvoie 0 si la somme est nulle, -1 si elle est négative et 1 si elle est positive (opérateur ternaire vu en classe inversée 05/10 slide 12). La somme est strictement positive dès la 3ième valeur lue, ce qui fait que la taille de vect est 3. Cette taille étant différente de celle de test_case, aucun message n'est affiché.

## 2) (2 pts) évaluation d'expression

Donner la valeur affichée par l'instruction cout. Montrer les priorités mises en œuvre par le langage C++ en soulignant les sous-expressions et en indiquant la valeur et le type des calculs intermédiaires.

```
cout << ((5/2 >= 2.5)+ not 0.0)%('C' - 'A') << endl ;[W]
cout << ((5/2 >= 2.5)+ not 0.0)%('F' - 'A') << endl ;[B]
cout << ((5/2 >= 2.5)+ not 0.0)%('D' - 'A') << endl ;[G]
cout << ((5/2 >= 2.5)+ not 0.0)%('E' - 'A') << endl ;[Y]
```



**3) (6 pts) Application de commerce en ligne avec erreur(s) sémantique(s):**

Le programme suivant compile correctement avec C++11.

Il doit afficher l'état des variables **monPanier** et **monStock** avant et après une suite de trois appels de la fonction **add_item**. Chaque appel à la fonction **add_item** essaie d'ajouter le **produit** fourni en argument au panier. Si le **produit** n'est pas en quantité suffisante dans le **stock**, aucun ajout n'est fait. Si la quantité est suffisante, le **Panier** et le **Stock** doivent être mis à jour

.

**3.1)** Quel est l'affichage réalisé par ce programme tel qu'il apparaît dans les pages suivantes ?[W : les autres variantes ont les mêmes articles mais avec des prix et des quantités différentes ; le nombre de ligne à afficher est le même pour toutes les variantes]

```
Panier de Sam
Total = 0 CHF

Etat du stock
Produit : Poire Prix: 2.5   qte: 2
Produit : Radis Prix: 1.5   qte: 5
Produit : Epis   Prix: 1.8   qte: 10

Poire   produit disponible
Radis  produit disponible mais quantité insuffisante
Epis    produite disponible

Panier de Sam
Total = 0 CHF

Etat du stock
Produit : Poire Prix: 2.5   qte: 2
Produit : Radis Prix: 1.5   qte: 5
Produit : Epis   Prix: 1.8   qte: 10
```

**3.2)** Y a-t-il une seule ou plusieurs erreurs sémantiques ? : (notion vue en série2 sur moodle)
**Il s'agit d'une erreur sémantique ayant un impact sur plusieurs lignes de code.**

Décrire la nature de chaque erreur
D'après la donnée la fonction add_item doit modifier le Panier et le Stock si certaines conditions sont remplies. Or les structures sto et pan sont **passées par valeur,** ce qui ne **permet pas leur modification à l'extérieur de la fonction add_item**. C'est pourquoi le panier reste vide avec un affichage de 0 CHF et le stock n'est pas modifiée après les appels de add_item.

**3.3)** Que faut-il modifier pour obtenir un fonctionnement correct du programme.
Faire un **passage par référence** des structures **sto** et **pan** pour pouvoir les modifier :
Ligne 25 sur première page du code

```
void add_item(string produit, int qte,Stock& sto,Panier& pan);
```

Ligne 2 sur seconde page du code

```
void add_item(string produit, int qte,Stock& sto,Panier& pan)
```

**3.4)** indiquer ce qui est modifié dans l'affichage du programme corrigé
Le second affichage du panier de Sam et du stock sont modifiés (un résumé des modifications est accepté):

**W=blanc, B=bleu, G=vert, Y=jaune, All = même réponse pour tous**

[W]
Panier de Sam
Produit : Poire Prix : 2.5 qte : 1
Produit : Epis Prix : 1.8 qte : 1
Total = 4.3 CHF

Stock : une unité de moins pour Poire et Epis, respectivement 1 et 9.

[B]
Panier de Sam
Produit : Radis Prix : 1.5 qte : 2
Produit : Epis Prix : 1.8 qte : 1
Total = 4.8 CHF

Stock : 2 unités de moins pour Radis et 1 de moins pour Epis, respectivement 3 et 9.

[G]
Panier de Sam
Produit : Poire Prix : 2.5 qte : 1
Produit : Radis Prix : 1.5 qte : 4
Total = 8.5 CHF

Stock : 4 unités de moins pour Radis et 1 de moins pour Poire, respectivement 1 et 1.

[Y]
Panier de Sam
Produit : Poire Prix : 2.1 qte : 1
Produit : Epis Prix : 2.8 qte : 1
Total = 4.9 CHF

Stock : une unité de moins pour Poire et Epis, respectivement 1 et 9.

```
1
2   #include <iostream>
3   #include <vector>
4   #include <string>
5
6   using namespace std;
7
8   struct Produit {
9       string nom;
10      float prix;
11  };
12
13  struct Panier {
14      string utilisateur;
15      vector<Produit> list;
16      vector<int> qte;
17      float total=0.0;
18  };
19
20  struct Stock {
21      vector<Produit> list;
22      vector<int> qte;
23  };
24
25  void add_item(string produit, int qte,Stock sto,Panier pan);
26  void actualise_total(Panier& pan);
27  void show(Produit prod);
28  void show(Panier pan);
29  void show(Stock sto);
30
31  int main()
32  {
33      Produit fruit ={"Poire",2.5};
34      Produit legume={"Radis",1.5};
35      Produit pain  ={"Epis", 1.8};
36
37      Stock monStock;
38      vector<Produit> list({fruit,legume,pain});
39      vector<int> qte({2,5,10});
40      monStock.list=list;
41      monStock.qte=qte;
42
43      Panier monPanier;
44      monPanier.utilisateur="Sam";
45
46      show(monPanier);
47      show(monStock);
48
49      add_item("Poire",1,monStock,monPanier);
50      add_item("Radis",6,monStock,monPanier);
51      add_item("Epis", 1,monStock,monPanier);
52
53      show(monPanier);
54      show(monStock);
55  }
56  // suite du code source en page suivante
```

```
1
2   void add_item(string produit, int qte,Stock sto,Panier pan)
3   {
4       cout << produit << "  ";
5       bool ok(0);
6       for(size_t i=0;i<sto.list.size();i++)
7       {
8           if (sto.list[i].nom==produit)
9           {
10              cout << "\tproduit disponible" ;
11              if (sto.qte[i]<qte)
12                  cout << " mais quantité insuffisante" << endl;
13              else
14              {
15                  cout << endl ;
16                  pan.list.push_back(sto.list[i]);
17                  pan.qte.push_back(qte);
18                  sto.qte[i]-=qte;
19                  ok=1;
20              }
21          }
22      }
23      if (ok) actualise_total(pan);
24  }
25
26  void actualise_total(Panier& pan)
27  {
28      float c=0.0;
29      for (size_t i=0;i<pan.list.size(); i++)
30          c+=pan.list[i].prix*pan.qte[i];
31      pan.total=c;
32  }
33
34  void show(Produit prod)
35  {
36      cout << "Produit: " << prod.nom << "\tPrix: " << prod.prix ;
37  }
38
39  void show(Panier pan)
40  {
41      cout << endl <<"Panier de " << pan.utilisateur << endl;
42      for (size_t i=0;i<pan.list.size();i++)
43      {
44          show(pan.list[i]);
45          cout << "\tqte:" << pan.qte[i] << endl;
46      }
47      cout << "Total = " << pan.total << " CHF"<< endl << endl;
48  }
49
50  void show(Stock sto)
51  {
52      cout << "Etat du stock" <<  endl;
53      for (size_t i=0;i<sto.list.size();i++)
54      {
55          show(sto.list[i]);
56          cout << "\tqte:" << sto.qte[i] << endl;
57      }
58      cout << endl ;
59  }
60
```

**4) (2 pts) fonction récursive**

```
1   #include <iostream>
2   using namespace std;
3   int f(int k); // [1]
4
5   int main()
6   {
7       cout << f(16) << endl;
8       cout << f(17) << endl;
9       return 0;
10  }
11
12  int f(int k)
13  {
14      if(k == 0 or k==1)
15          return 1;
16       return k*f(k-1);
17  }
```

L'exécution de ce programme produit l'affichage suivant :
**2004189184**
**-288522240**
**E**st-ce correct ? **[All]** Non
**J**ustifier votre réponse :**[All]** la fonction f(k) calcule la factorielle de k qui est toujours positive.
on obtient une valeur négative à cause de la violation du domaine couvert obtenue pour k valant 17. Le résultat est plus grand que le max des nombres positifs en complément à 2 et il est interprété comme un nombre négatif car le bit de signe est à 1.
(pas demandé car pas de calculatrice) même f(16) est plus petite que sa valeur correcte de 20922789888000

**5) (3pts) programme mystère**

**5.1)** Indiquer le résultat affiché par ce programme : solutions page suivante

```
1   #include <iostream>
2   #include <vector>
3   #include <string>
4
5   using namespace std;
6   const vector<vector<string>> tab={
7       {"","M","MM","MMM"},
8       {"","C","CC","CCC","CD","D","DC","DCC","DCCC","CM"},
9       {"","X","XX","XXX","XL","L","LX","LXX","LXXX","XC"},
10      {"","I","II","III","IV","V","VI","VII","VIII","IX"}
11  };
12
13  int main()
14  {
15       int n(1492),d(1000);
16
17       for(size_t i(0) ; n ; ++i, n %= d, d /=10)
18            cout << tab[i][n/d];
19       cout << endl;
20       return 0;
21  }
```

[1] Nous avons compté juste pour ceux qui ont remarqué qu'il manquait cette déclaration mais ça n'était pas le but de l'exercice car le code ne compile pas en l'absence de cette déclaration. La version publique de l'examen contient cette déclaration de façon à se concentrer sur le problème de violation du domaine couvert par le type **int** et qui est décrit dans la solution ci-dessus.

**5.1)** Indiquer le résultat affiché par ce programme :
[W] n vaut 1492 : l'affichage de ce programme est : MCDXCII
[B] n vaut 1291 : l'affichage de ce programme est : MCCXCI
[G] n vaut 1936 : l'affichage de ce programme est : MCMXXXVI
[Y] n vaut 1459 : l'affichage de ce programme est : MCDLIX

**5.2)** Quel est l'intervalle [min, max] des valeurs de n pour lesquelles ce programme est correct :

[All] On entend par « valeurs correctes » les valeurs de **n** qui produisent des **indices valides pour accéder aux éléments de tab**, c'est-à-dire entre 0 et 3999.

**5.3)** Indiquer l'affichage produit par les valeurs min et max :

[All] Lorsque n vaut 0 on n'entre même pas dans la boucle for, donc seulement un passage à la ligne.

Lorsque n vaut 3999 le programme affiche : MMMCMXCIX