EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communication
Cours de programmation à la section SV
Sam J.

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PROGRAMMATION I

Corrigé Examen Semestre I

Exercice 1 : Conception OO et programmation (45 points)

Il existe plusieurs façons de mettre en oeuvre une conception correcte.
Notez que seuls les prototypes étaient demandés dans la partie 1.
Un corrigé possible est fourni dans les fichier sources `jardin.cc`

Exercice 2 : Questions sur les concepts (32 points)

1. (a) Incorrecte, on ne peut afficher tout un vector d'un coup et `tab[9]` est un `vector` de 3 entiers.
   (b) Incorrecte, pour les mêmes raisons que pour le point précédent et parcequ'il y a débordement d'indice.
   (c) Correcte et affiche 3.
   (d) Correcte et affiche 1.
   (e) Incorrecte car il y a débordement d'indice sur la première dimension.
   (f) Incorrecte car `tab[5][0]` est un entier et la fonction `size` ne peut s'y appliquer.
2. `typedef int* Truc;`
3. (a) Oui car elle a le même nom et diffère de la première par la liste et le type des arguments.
   (b) `bool grand(string s){return s.size() > 250;}`
   (c) `void grand(int, int)` (le type de retour ne fait pas partie de la signature)
4. la première boucle parcourt le tableau en essayant de la modifier au travers d'une variable `v` qui n'est pas une référence à l'entrée correspondante du tableau. Il faut remplacer `auto` par `auto&` dans cette boucle.
5. Le programme affiche :

   ```
   est-ce ainsi
   que les hommes vivent ?
   ```

   Justification: la fonction `f` manipule une copie de son premier argument `s1` (passage par valeur). Les modifications faites sur `s1` dans `f` ne se répercutent pas à l'extérieur de la fonction. La chaîne passée en argument reste inchangée. Par opposition, `g` utilise un passage par référence pour son premier argument et celui-ci est bien modifié par la fonction de sorte que la modification reste visible après que la fonction ait fini son exécution.
6. La fonction `f` a pour type de retour `void`. Elle ne retourne donc aucune valeur qui peut être affichée sur `cout`. Correction:

   ```
   cout << "Mon nom est << flush;
   f("personne");
   ```

7. Une solution possible:

```
int f(vector<int> v)
{
  if (!v.empty()) {
    int sum(0);
    for (auto n : v) {
      sum += n;
    }

    if (sum < 0){
      throw 0; // code d'erreur
    }

    return sum;
  }
}

int main()
{
  vector<int> v1({ -15, 3, -9, -2, 5 });
  vector<int> v2({ 2, 5 });
  try{
    cout << "somme1 : " << f(v1) << endl;
    cout << "somme2 : " << f(v2) << endl;
  }

  catch(int code)
  {
    cout << "calcul d'une somme négative" <<endl;
  }

  return 0;
```


Exercice 3 : Déroulement de programme [20 points]

— La fonction `e` permute les deux entrées d'indices `a` et `b` dans un vecteur de `D` : la structure initialement à la position `a` ira à la position `b` et celle initialement à la position `b` ira à la position `a`. Le procédé est le même que celui utilisé en TP par la fonction `echange` (utilisation d'une variable intermédiaire pour éviter l'écrasement de la deuxième valeur à déplacer).

— la fonction `op` trie le vecteur `d` par ordre croissant du premier champ de chaque entrée (en fait trie des dates par ordre croissant des jours). Elle procède comme suit : pour tout indice possible `i`, examiner toutes les entrées positionées après `i`. Si une de ces entrées, `d[j]` a son champ `j` plus petit que celui de `d[i]` on, permute `d[i]` et `d[j]` (au moyen la fonction `e`).

— la fonction `a` : affiche pour chaque entrée du vecteur `d`, ses 3 champs séparés par `"/"`. L'affichage du dernier champ est précédé de l'affichage de `20` (en fait sous la forme de dates !).

— affichage produit :

```
Init
Process
1/1/2017
4/1/2017
25/1/2017
30/1/2017
```