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 de programme et programmation (65 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 `rennes.cc`

Exercice 2 : Questions sur les concepts (30 points)

1. (a) Si l'utilisateur ne saisit pas une valeur attendue d'emblée (à savoir `pluie` ou `soleil`), le programme rentre dans une boucle infinie. En effet la condition de continuation de la boucle porterait alors sur la variable `answer` accessible à cette portée (c'est à dire celle définie à la ligne 2) et qui contient la mauvaise valeur. La variable `answer` de portée locale au corps de la boucle prend des nouvelles valeurs à la lecture mais n'est jamais utilisée et ne permet donc pas de sortir de la boucle.

   (b) La répétition peut être évitée en utilisant la bonne structure de contrôle, c'est à dire un `dowhile`

```
string answer;
do {
     cout << "Quel temps fait-il ?<< endl;
     cin >> answer;
   }
while(answer != "soleil" and answer != "pluie");
```

2. Le programme affiche :

```
9
```

   L'évaluation de l'expression affichée par la ligne 15 commence par l'appel de `f(i)`, `i` vaut alors 5 mais est passé *par référence* à la fonction `f`. la ligne 2 met donc la valeur 1 dans i, la ligne 3 l'incrémente (i vaut alors 2 et c'est cette valeur 2 qui est aussi retournée par l'appel `f(i)`. Est ensuite évalué `g(i)`, avec `i` valant 2. `i` est cette fois passé *par valeur* ce qui veut dire que les lignes 8 à 10 vont travailler sur une copie de `i`. l'appel `g(i)` retourne 5, mais la valeur de `i` reste 2. L'expression `f(i) + g(i) + i` retourne donc `2 + 5 + 2` c'est à dire `9`.

3. (a) lorsqu'un traitement a besoin d'identifier un indice, il est préférable d'itérer avec un `for` classique (qui va de toute façon faire progresser l'indice)

   (b) L'algorithme itère sur toutes les valeurs du tableau alors qu'il suffirait de s'arrêter dès que l'on a trouvé la première valeur paire. De plus il est erroné. Il incrémente en effet l'indice `i` tant qu'il n'a pas trouvé le premier nombre pair, mais il l'incrémente aussi à chaque valeur impaire trouvée après la première valeur paire, ce qui ne donne pas le résultat escompté.

   (c) Le programme devrait plutôt être :

```
for (size_t i(0), i < t.size(); ++i){
    if (t[i]%2 == 0)
        return i;
}
return -1;
```

4. (a) Le programme affiche

```
layal
5
```

   (b) la ligne 15 déclare un variable `s` de type `S` dont le champs s vaut la chaine `"royal` et le champ `c` vaut zéro. la variable `s` est passée par référence à la fonction `f` dans la boucle de la ligne 16. `f(i)` remplace le ième élément de `s.s` par celui à la position miroir (`f(0)` remplace `s.s[0]` par `s.s[4]`, `f(1)` remplace `s.s[1]` par `s.s[3]` etc). Lorsqu'on dépasse la milieu de la chaîne chaque valeur de `s.s[i]` a déjà la même valeur que son miroir et la chaîne ne change plus. Chacun des 5 appels à `f` cause l'incrémentation de `s.c` lequel ne va pas suffisamment loin pour que l'on passe par les lignes 6 et 7.

5. (a) il faut spécifier le nom du fichier de sortie dans la ligne 7, par exemple :

```
ofstream sortie("output.txt");
```

   Il faut également fermer le flot avant la ligne 9 :

```
close(sortie);
```

(b) non, car les flots ne peuvent pas être passés par valeur.

6. (a) Le code affiche :

```
g(string)
f(string, string)
magic wonderland land
```

Justification : la ligne 18 cause l'appel de la fonction `g` de la ligne 6 avec le paramètre `"land"`. La ligne 8 affiche `g(string)`. la ligne 10 invoque ensuite la fonction `f` de la ligne 1 en lui passant comme paramètre `s` qui vaut `"land"` et une seconde chaîne `"wonder"`. La ligne 11 affiche `f(string, string)`. La ligne 4 lance une exception sous la forme d'une `string` valant `"magic wonderland"` (la variable `s` de la ligne 4 est la constante globale de la ligne zéro. le bloc réceptif à cette exception est celui commençant à la la ligne 11 lequel relance un exception sous la forme de la chaîne `"magic wonderland land"`. Cette exception est rattrapée à la ligne 21, ce qui affiche `"magic wonderland land"`.

(b) si l'on inverse les lignes 18 et 19, le programme affiche:

```
f(string, string)
magic lost abbacus
```

Justification : l'appel `f("abbacus", "lost")` cause l'affichage de `f(string, string)` puis provoque le lancement d'une exception prenant la forme de la chaîne `"magic lost abbacus"`, cette exception est rattrapée en ligne 20 ce qui provoque l'affichage de `"magic lost abbacus"`. L'appel à `g` est court-circuité.

Exercice 3 : Déroulement de programme [20 points]

Le programme affiche :

```
Orange Black Blue Yellow
*****
Black Blue Orange Yellow
*****
Yellow
Orange
Blue
Black
*****
Colorless world
```

Justification : La ligne 48 déclare un tableau dynamique `t` d'objets de type `Color` (structure de données dont le premier champ est un entier et le second un pointeur sur une chaîne de caractères). Les lignes 49 à 52 remplissent ce tableau au moyen de quatre valeurs dont les champs `n` sont des pointeurs vers des chaînes de caractères dynamiquement allouées. La ligne 53 appelle le fonction de la ligne 19, ce qui affiche les valeurs pointées par les champs `n` des quatre valeurs du tableau, donnant :

```
Orange Black Blue Yellow
```

La ligne avec `*****` s'affiche ensuite, puis la ligne 55 s'exécute ce qui cause l'appel de la fonction `g` de la ligne 10. Le tableau `t` est passée par référence à cette fonction. `g` trie le tableau passé en paramètre par valeur croissante des champs `w`. A la sortie de `g`, le tableau `t` du `main` est donc trié et son affichage à la ligne 56 donne:

```
Black Blue Orange Yellow
```

La ligne avec `*****` s'affiche à nouveau, puis s'exécute la boucle de la ligne 58 qui va appeler 4 fois la fonction `f` de la ligne 30 en lui passant le tableau `t` en paramètre par référence. La fonction `f` affiche la chaîne de caractère de l'entrée du tableau ayant la plus grande valeur dans son champ `w`. Ce champs est ensuite remplacé par -1. La première fois que `f` est appelée elle affichera donc `Yellow`, l'entrée du tableau correspondante n'aura donc ensuite plus la plus grande valeur de `w`, ce sera donc la seconde plus grand valeur qui s'affiche à savoir `Orange` et ainsi de suite. La boucle de la ligne 58 affiche donc les entrées de `t` de la plus grande à la plus petite (le critère de comparaisom étant le champs `w`). Ceci donne donc :

```
Yellow
Orange
Blue
Black
```

S'affiche ensuite à nouveau `*****`. La ligne 62 vide ensuite le tableau et l'appel de la ligne 63 va causer l'exécution des lignes 21 à 23, ce qui affiche :

```
Colorless world
```

A propos de la gestion de la mémoire: les chaînes de caractères dynamiquement allouées en lignes 49 à 52 ne font jamais l'objet d'un delete. Il faudrait ajouter le code suivant avant la ligne 62 :

```
for (auto& val : t){
 delete val.n;
}
```