INVENTEURS DU MONDE NUMÉRIQUE

CONCOURS EXTERNE D'INGENIEUR DE RECHERCHE
DI1

BAP E – INFORMATIQUE ET CALCUL SCIENTIFIQUE

Ouvert au titre de 2012
Arrêtés du 29 mars 2012 parus au JO du 8 avril 2012
Epreuve écrite
Note sur 20 – Coefficient 3 – Durée trois heures

le 19 juin 2012 de 14h à 17h

Le candidat peut traiter les questions dans l'ordre de son choix.
Veillez à respecter l'anonymat dans les réponses.

Une attention particulière sera portée à la qualité de la présentation, à l'orthographe et à la grammaire.

Module A : (total : 7 points)

Exercice A1 : **(3 points)**

Pour représenter un graphe, par exemple dans le langage C, on utilise la structure de données suivante :

```
# define  n_max …       /* nombre maximum de sommets */
typedef   int sommet;   /* indice des sommets : de 0 à n_max-1 */

/* représentation par matrice d'adjacence */
typedef struct {
int a[n_max][n_max];    /* matrice carrée  */
int n ;                 /* nombre de sommets du graphe */
} graphe_m ;
```

On considère que le graphe est pondéré, c'est-à-dire que chaque arête du graphe se voit affectée d'un « poids » qui est représenté par un entier supérieur à zéro. Dans la matrice d'adjacence **a** de la structure de données présentée, une valeur de zéro correspond à une absence d'arête entre les deux sommets correspondants ; une valeur supérieure à 0 correspond à une arête dont le poids est inscrit dans la matrice.

On vous demande de coder en langage C, Java ou C++, en adaptant cette structure de données au langage choisi, une fonction construisant un arbre de recouvrement de poids minimal d'un graphe passé en paramètre, par l'algorithme de Prim.

On rappelle l'algorithme de Prim :
Soit un graphe G, connexe, non orienté et pondéré, dont l'ensemble de sommets est V et l'ensemble d'arêtes est E.

1. On initialise l'ensemble de sommets (de l'arbre de recouvrement) X avec un sommet au hasard.
2. On initialise l'ensemble d'arêtes (de l'arbre de recouvrement) T à l'ensemble vide.

3. On cherche (a,b) l'arête de E de plus petit poids et qui relie un sommet a de X à b qui n'est pas dans X, on ajoute alors cette arête à T, on ajoute le sommet b à l'ensemble X.
4. Si X n'est pas égal à l'ensemble V, alors on revient à l'étape 3, sinon on va à l'étape 5.
5. L'ensemble T constitue l'arbre de recouvrement de poids minimal.

Vous préciserez bien les structures de données complémentaires que vous utilisez avant de décrire la fonction calculant cet arbre par la méthode de Prim. Vous indiquerez ensuite le coût asymptotique de cet algorithme en justifiant votre réponse.

Exercice A2 : **(1,5 points)**

a) Il s'agit de paralléliser le plus efficacement possible la boucle suivante en vue d'une exécution sur une machine à mémoire partagée dotée de P cœurs. On suppose évidemment que tous les appels à la fonction f sont indépendants. Vous êtes libre d'utiliser l'outil de programmation parallèle de votre choix (OpenMP, Intel TBB, etc.) Veillez à bien expliquer l'approche choisie.

```
for (i=0 ; i < N ; i++)
      s += f(i) ;
```

1. En supposant que le temps de calcul de `f(i)` ne dépend pas de la valeur de `i` (i.e. le temps de calcul est constant) ;
2. En ne pouvant rien supposer sur le temps de calcul de `f(i)` qui est arbitraire.

b) Décrivez l'architecture générale d'un accélérateur graphique (GPU). Expliquez à quoi ressemble un programme destiné à s'exécuter sur un tel accélérateur : quel est le paradigme de programmation ?

Exercice A3 : **(2,5 points)**

La gestion de la mémoire dynamique (allocation/libération) est depuis toujours un problème important dans l'implémentation des langages de programmation. Deux approches sont possibles : i) laisser cette gestion à la charge du programmeur ou ii) gérer la mémoire automatiquement.

a) Dressez une liste des inconvénients de chacune des deux approches.

Que la gestion de la mémoire dynamique soit manuelle ou automatique, on utilise un système de ramasse-miettes (ou *garbage collector*).

b) Quel est le rôle du ramasse-miettes ?

Il existe un grand nombre de stratégies pour gérer la mémoire dynamique (encore appelée le tas) au moyen d'un ramasse-miettes. Le principe général est toujours le même :

On considère le tas comme un graphe, dont les sommets sont les blocs (objets, tableaux...) et les arêtes sont les pointeurs. Les blocs auxquels le programme peut accéder directement (parce qu'il possède leur adresse) sont les racines du graphe. Tout objet qui n'est pas accessible depuis une racine en parcourant le graphe est mort (garbage) et peut être libéré. De temps en temps, on examine le graphe pour déterminer quels objets sont accessibles. On libère les autres.

Voici 3 stratégies qui sont employées :

1. **Comptage de références :** Elle consiste à stocker dans chaque bloc un compteur du nombre de pointeurs vers ce bloc. Lorsque ce compteur devient 0, le bloc est libéré.
2. **Mark and sweep :** On parcourt le graphe depuis les racines en marquant les sommets accessibles ; puis on *balaie* le tas linéairement en libérant tous les blocs qui n'ont pas été marqués et on enlève la marque pour les autres.
3. **Stop and copy :** On recopie la partie du graphe accessible dans une autre partie du tas. Après recopie, les blocs seront contigus. Le tas est donc divisé en deux zones. L'une des deux n'est pas utilisée jusqu'à la copie. Ensuite, on intervertit le rôle des deux zones.

c) Quel est selon vous le désavantage majeur de chacune de ces stratégies ?

Module B : (total : 5,5 points)

Ce module utilise les documents en Annexe A

Toutes vos réponses sont à rédiger en anglais

Exercice B1 : **(4,5 points)**

Vous devez participer à la réalisation de l'application « *clheuredyaller* » (c'est l'heure d'y aller).

L'utilisateur doit indiquer à l'application l'endroit où il se trouve, l'endroit où il doit se rendre et l'heure limite d'arrivée qu'il souhaite. En retour, l'application indique l'heure limite à laquelle il doit partir. Il peut utiliser l'application sur un téléphone ou un ordinateur. Dans une première phase, l'application s'appuie sur les transports en commun ou un déplacement à pied.

Dans le cadre "opendata", plusieurs villes proposent des interfaces (voir l'annexe A).

a) Quelles API vous semblent nécessaires (incontournables) ou plus adaptées ?

b) Quelles API peuvent apporter un réel avantage ?

c) Sachant que vous utilisez cet environnement externe, décrivez les difficultés à traiter.

d) Sachant que vous envisagez une première expérimentation sur téléphone, proposez une stratégie pour la mettre au point.

Exercice B2 : **(1 point)**

Identifiez 2 risques relatifs à l'utilisation des jeux de données ou API externes. Pour chaque risque, proposez une posture ou solution de contournement.

Module C : (total : 7,5 points)

Exercice C1 : **(3 points)**

Vous devez réaliser ***un jeu du pendu*** consistant à trouver un mot en devinant les lettres qui le composent. La classe ***Jeu*** permet de commencer et terminer une partie et de définir les options pour une partie (nombre de tentatives permises, temps limité ou non, niveau de difficulté des mots à trouver). La classe ***Partie*** indique le nombre maximal de tentatives permises et le nombre de tentatives effectuées par le joueur pour la partie en cours. La partie en cours évolue en fonction des coups effectués par le joueur (lettres choisies).

Lorsqu'un caractère est saisi par le joueur, il faut effectuer les vérifications suivantes :

- le caractère est une lettre de l'alphabet,
- la lettre fait partie du mot,
- la lettre n'a pas déjà été choisie auparavant.

La mise en place de ce jeu est effectuée par deux développeurs dont un spécialiste en IHM.

En suivant *le patron de conception, ou le pattern, MVC (Modèle, Vue, Contrôleur)* :

a) Décrivez les cahiers des charges que vous donnerez à chacun des développeurs. Citez les classes principales et leurs interfaces (API). L'IHM peut être schématisée graphiquement si nécessaire.

b) Indiquez, pour la partie modèle, les informations à remonter en utilisant les événements et illustrez ceci sur un cas.

c) Indiquez, pour chaque vérification mentionnée ci-dessus, à quel niveau l'effectuer : modèle, vue ou contrôleur. Expliquez votre choix.

Exercice C2 : **(0,5 point)**

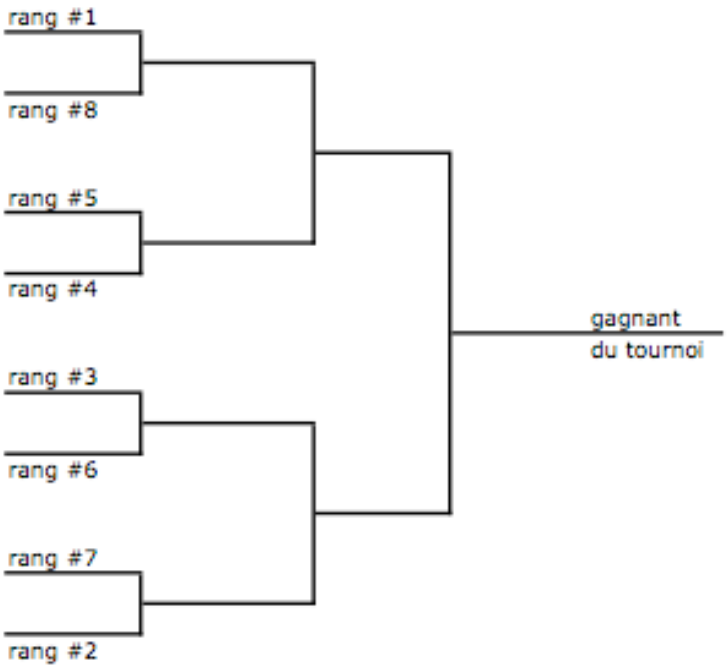
Vous venez d'être recruté par la société *VolFacile* qui souhaite lancer son nouveau service de réservation pour les abonnés d'un vol régulier. Ils s'enregistrent une fois pour l'année et reçoivent automatiquement sur leur mobile leur carte d'embarquement. Ce service nécessite de s'inscrire en donnant son nom, son prénom et le(s) vol(s) concerné(s) : ville de départ, ville de destination, horaire, jour de la semaine et périodicité (hebdomadaire, bimensuel ou mensuel). L'inscription renvoie soit un accusé de réception négatif en cas d'erreur, soit un identifiant unique pour la réservation. Une fois l'inscription effectuée, les clients sont automatiquement enregistrés. Les clients peuvent envoyer une requête d'annulation ponctuelle ou définitive. Pour une annulation ponctuelle, le client envoie l'identifiant et le nombre de vols consécutifs concernés par l'annulation. En retour, il reçoit une confirmation, accompagnée d'une éventuelle pénalité financière.

a) On vous demande de mettre en place le serveur qui permet de répondre à ces besoins. Décrivez **en 10 lignes maximum** les technologies que vous connaissez qui vous permettent de mettre en place une architecture client-serveur et la mise en œuvre sous jacente.

b) Donnez une description du protocole d'application.

Exercice C3 : **Cet exercice utilise les documents en Annexe B** **(3,5 points)**

Lors de l'organisation d'un tournoi de basket à élimination directe, si les équipes possèdent un rang de classement avant le début du tournoi, il existe une méthode d'organisation de tournoi qui augmente les chances de voir les meilleures équipes (au sens du rang avant le début du tournoi) se rencontrer dans les demi finales et la finale. Le principe est de faire se rencontrer l'équipe du meilleur rang (rang #1) et l'équipe la moins bien classée, l'équipe de rang #2 avec l'avant-dernière et ainsi de suite. Ceci constitue la première étape du tournoi. Le schéma suivant montre un tournoi de 8 équipes.



Le vainqueur de chaque match rencontre celui d'un autre match comme indiqué sur le schéma précédent. On obtient donc une succession d'appariements pour chaque phase du tournoi jusqu'à la finale.

Une application pour gérer les tournois de basket (qui peut probablement être utilisée pour d'autres sports) pourrait manipuler des objets pour représenter les équipes (***Team***), les matchs (***Game***) et les tournois (***Tournament***). Un tournoi a un ensemble de matchs et un ensemble d'équipes. Une fois que le match a été joué, l'équipe gagnante (***winner***) avance à la phase suivante du tournoi.

Considérez les classes définies dans l'annexe B.

a) Quel est la forme canonique dans une classe en C++ ? Quelles classes sont canoniques dans le code ?

b) Donnez l'implémentation de la classe ***Game.***

c) Est-il possible d'implanter ***Tournament::run*** plus facilement ? Nommez la technique que vous utiliserez et donnez une implantation.

d) Un patron de conception (design pattern) très connu est caché dans le code fourni. Nommez ce patron et changez le nom de la classe ***X*** pour qu'il corresponde à ce patron. Quelle est la particularité de l'usage de ce patron dans ce code ?

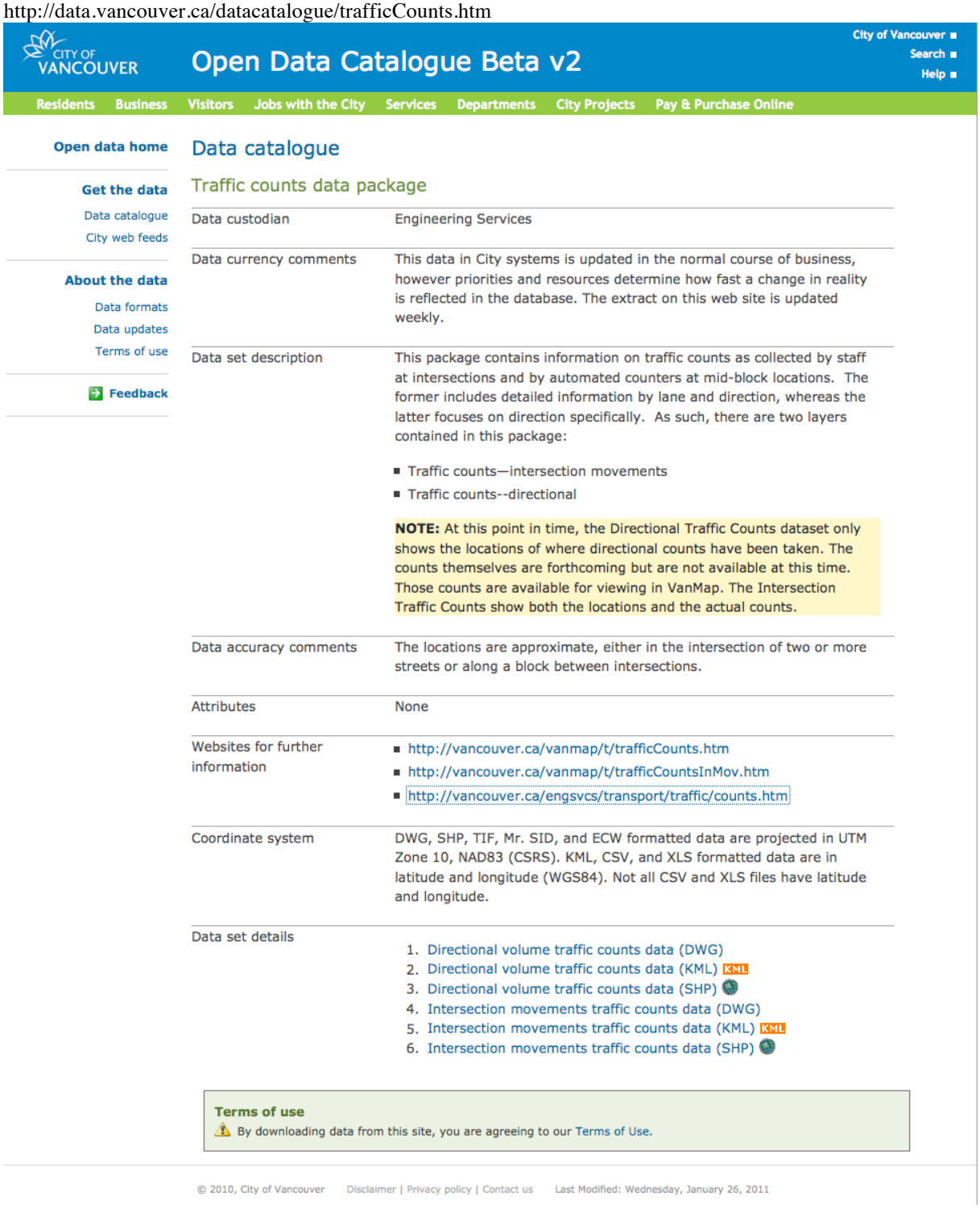
e) Donnez une implantation des classes ***Subject*** et ***X*** (renommées par vos soins) pour refléter le patron dans la question précédente. Expliquez les particularités de votre implantation.

Exercice C4 : **(0,5 point)**

A quel patron de conception (*design pattern*) pensez-vous quand vous voulez mettre en œuvre une et une seule instance d'une classe ? Donnez l'implantation et expliquez comment ceci assure que seule une instance est possible.

ANNEXE A à utiliser pour le module B

http://data.vancouver.ca/datacatalogue/trafficCounts.htm



CITY OF VANCOUVER

Open Data Catalogue Beta v2

City of Vancouver ■ Search ■ Help ■

Residents | Business | Visitors | Jobs with the City | Services | Departments | City Projects | Pay & Purchase Online

Open data home

Get the data
Data catalogue
City web feeds

About the data
Data formats
Data updates
Terms of use

Feedback

Data catalogue

Traffic counts data package

Data custodian	Engineering Services
Data currency comments	This data in City systems is updated in the normal course of business, however priorities and resources determine how fast a change in reality is reflected in the database. The extract on this web site is updated weekly.
Data set description	This package contains information on traffic counts as collected by staff at intersections and by automated counters at mid-block locations. The former includes detailed information by lane and direction, whereas the latter focuses on direction specifically. As such, there are two layers contained in this package: ■ Traffic counts—intersection movements ■ Traffic counts--directional **NOTE:** At this point in time, the Directional Traffic Counts dataset only shows the locations of where directional counts have been taken. The counts themselves are forthcoming but are not available at this time. Those counts are available for viewing in VanMap. The Intersection Traffic Counts show both the locations and the actual counts.
Data accuracy comments	The locations are approximate, either in the intersection of two or more streets or along a block between intersections.
Attributes	None
Websites for further information	■ http://vancouver.ca/vanmap/t/trafficCounts.htm ■ http://vancouver.ca/vanmap/t/trafficCountsInMov.htm ■ http://vancouver.ca/engsvcs/transport/traffic/counts.htm
Coordinate system	DWG, SHP, TIF, Mr. SID, and ECW formatted data are projected in UTM Zone 10, NAD83 (CSRS). KML, CSV, and XLS formatted data are in latitude and longitude (WGS84). Not all CSV and XLS files have latitude and longitude.
Data set details	1. Directional volume traffic counts data (DWG) 2. Directional volume traffic counts data (KML) KML 3. Directional volume traffic counts data (SHP) 4. Intersection movements traffic counts data (DWG) 5. Intersection movements traffic counts data (KML) KML 6. Intersection movements traffic counts data (SHP)

Terms of use

By downloading data from this site, you are agreeing to our Terms of Use.

 Disclaimer | Privacy policy | Contact us Last Modified: Wednesday, January 26, 2011

http://vancouver.ca/vanmap/t/trafficCounts.htm

Start VanMap
About the Data
List of Layers
Documentation Index
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Search
Go

Traffic Counts

Layer name: Traffic Counts - Directional Volume

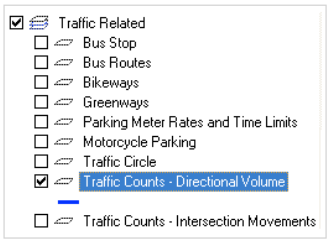
Group name: Traffic Related

Scale: Always

Data Currency Status: Updated to include 2007 and previous automated counts. It will be updated as required.

Dept/branch: Engineering, Traffic Management

Definition: These reports summarize the automated traffic volume counts completed by the Transportation Division, City of Vancouver Engineering Services.



Automatic counts are conducted by placing hoses on the roadway, usually for a 24-hour period, and recording the number of vehicles traveling in a specific direction. The counts are done mid-block between two intersections to give a total for each direction of the street.

To access an auto-count report, double-click the highlighted street segment when you are in VanMap or access the Auto Counts from the application list box.

Note: For blocks with multiple street segments only one segment is highlighted but the count is for the entire block, and the highlighted segment does not reflect the actual location of the traffic counter.

[top]

http://vancouver.ca/vanmap/t/trafficCountsInMov.htm

Start VanMap
About the Data
List of Layers
Documentation Index
A B C D E F G H I
J K L M N O P Q R
S T U V W X Y Z
Search
Go

Traffic Counts - Intersection Movements

Layer name: Traffic Counts - Intersection Movements

Group name: Traffic Related

Scale: Always

Data Currency Status: Data is updated on an ongoing basis in Oracle Spatial. These changes are reflected immediately in VanMap.

Dept/branch: Engineering, Traffic Management

Definition: These reports summarize the manual traffic volume counts completed by the Transportation Division, City of Vancouver Engineering Services. Each location counted is represented by an Intersection Traffic Flow Diagram. The computer program that creates the flow diagrams calculates the A.M. and P.M. maximum ('peak') hour for the intersection.

Traffic volumes for the A.M. maximum hour, the P.M. maximum ('peak hour') and each corresponding two-hour 'rush period' are totaled by lane and printed to the right of each Flow Diagram. All motor vehicles (including trucks and buses) that enter the intersection are included in these volumes.

Manual traffic counts are conducted at intersections from 7-9am and 3:30-5:30pm. Volumes at signalized and non-signalized intersections are recorded in five-minute intervals, which are summarized into flow diagrams. Pedestrian movements are recorded at the crosswalks for these intersections (e.g. 'N' x-walk records pedestrians who cross the 'north leg' of the intersection during the 'peak' hour). Bicycle movements are recorded from the leg with which they originate.

To access a Traffic Count report, double-click the icon  when you are in VanMap.

[top]

http://data.vancouver.ca/dataFormats.htm

Open data home

Get the data
Data catalogue
City web feeds

About the data
Data formats
Data updates
Terms of use

Feedback

Data formats

Eventually we plan to make each type of data available in multiple formats; however, it will take some time to get there. Formats such as SHP, DWG, KML and others, where appropriate, will be provided. Not all data will be available in all formats and other formats may be added to this list.

- **CSV/XLS:** This is a basic text format that can be opened by many applications including Excel and other spreadsheet applications. Read about CSV format on WikiPedia Read about XLS on WikiPedia
- **DWG:** DWG is a "de facto GIS standard" format native to Autodesk design and GIS software. It can be consumed by a wide variety of Autodesk and non-Autodesk software packages. Read about DWG format on WikiPedia
- **ECW:** ECW is a format used for geo-referenced raster images that is highly compressed. Read about ECW on WikiPedia Read about image formats on WikiPedia
- **GeoJSON:** GeoJSON is a lightweight text-based open standard designed for human-readable data interchange. GeoJSON is a version of JSON that focuses on geographically-based data. Read about GeoJSON in Wikipedia Read about JSON in Wikipedia
- **JSON:** JSON is a lightweight text-based open standard designed for human-readable data interchange. Read about JSON in Wikipedia
- **KML:** KML is a mark-up language developed by Keyhole Inc. and subsequently acquired by Google to display data in Google Maps, Google Earth and similar applications. KML is an Open Geospatial Consortium standard. Read about KML format on WikiPedia
- **MrSID:** MrSID (pronounced "Mister SID") is a format used for geo-referenced raster images that is highly compressed. The MrSID version of the facetized orthophotos are only about 5% of the size of their GeoTiff versions. Read about MrSID format on WikiPedia Read about image formats on WikiPedia
- **SHP:** SHP (pronounced "shape") is a "de facto GIS standard" format native to ESRI GIS software. It can be consumed by a wide variety of ESRI and non-ESRI software packages. Read about SHP format on WikiPedia
- **Web feed:** A web feed (or news feed) is a data format used for providing users with frequently updated content. Content distributors syndicate a web feed, thereby allowing users to subscribe to it. Read about Web feed on Wikipedia

A note about projections:

DWG, SHP, TIF, Mr. SID and ECW formats are projected in UTM zone10, NAD83 (CSRS).

KML and CSV/XLS formats are projected in latitude, longitude (WGS84); this is relavent for CSV/XLS files only if they contain latitude and longitude.

 Disclaimer | Privacy policy | Contact us Last Modified: Thursday, January 12, 2012

http://data.vancouver.ca/termsOfUse.htm

Terms of use associated to Vancouver's Open Data Catalogue Beta v2

Introduction

Please note that by accessing the datasets, you agree to these Terms of Use, which are intended to protect and promote the City's commitments to open data and ensure that the recipients of these datasets give back to the community the benefits they derive from these datasets.

If you have any questions or comments about these Terms of Use, please feel free to contact us at opendata@vancouver.ca.

Your Open Licence to the Datasets

The City of Vancouver (City) now grants you a world-wide, royalty-free, non-exclusive licence to use, modify, and distribute the datasets in all current and future media and formats for any lawful purpose. You now acknowledge that this licence does not give you a copyright or other proprietary interest in the datasets. If you distribute or provide access to these datasets to any other person, whether in original or modified form, you agree to include a copy of, or this Uniform Resource Locator (URL) for, these Terms of Use and to ensure they agree to and are bound by them but without introducing any further restrictions of any kind.

Giving City Credit

Although you are not required to credit the City for each use or reproduction of the datasets, you are entitled to do so and encouraged to conspicuously announce that these datasets are publicly available from the City under these Terms of Use. Upon the request of the City, you may be required to remove a credit from future uses or reproductions should the City decide that such credit is not in the public interest.

Future Changes to Datasets/Terms of Use

The City may at any time and from time to time add, delete, or change the datasets or these Terms of Use. Notice of changes may be posted on the home page for these datasets or this page. Any change is effective immediately upon posting, unless otherwise stated.

Compliance With Law Your Responsibility

You assume sole responsibility for your use and reproduction of the datasets complying with all applicable laws and industry standards.

No Warranty with Datasets

You get NO WARRANTIES, none of any kind. By this, we mean, for example (but without limiting the total intended scope of the preceding sentence), (1) that while reasonable efforts have been made in preparing these datasets for use by you, the City cannot give any promises as to the completeness, currency, or accuracy of the datasets nor that access will be continuous, (2) the City cannot make any promise that the datasets are free and clear of any possible third party copyright, moral rights, or other claim, (3) the datasets have been modified from their original source, as data initially generated by the City for its internal uses, and (4) all data visualizations on maps are approximate and include only records that can be mapped.

You accept these datasets on an "as-is, where is" basis and agree to use them at your own risk.

Exclusion of Liability

You agree that you will not and cannot sue the City for anything which the City does or does not do (even if intentional or negligent) in connection with the datasets and your use or inability to use them. Without

limiting the general scope of the preceding sentence, this means that the City and its agents are not liable on any legal theory or basis for any direct, incidental, indirect, special, punitive, exemplary, or consequential damages or losses, including without limitation, loss of revenue or anticipated profits, loss of goodwill, loss of business, loss of data, computer failure or malfunction, or any other damages or losses.

Liability for Not Complying with Terms of Use
If, as a result of your breach of these Terms of Use, the City gets sued or is required to pay someone money, you agree to protect the City and reimburse the City for everything which you cause the City to suffer. This means that you agree to defend, indemnify, and hold harmless the City and all of its agents from any and all liabilities incurred in connection with any claim arising from any breach by you of these Terms of Use, including reasonable legal fees and costs. You agree to cooperate fully in the defense of any such claim. The City reserves the right to assume, at its own expense, the exclusive defense and control of any matter otherwise subject to indemnification by you. You agree not to settle any matter without the written consent of the City.

Cancellation for Non-Compliance
The City may, in its sole discretion, cancel or suspend your access to the datasets without notice and for any reason, including anything which the City, in its sole discretion, believes is a breach of these Terms of Use or is otherwise unlawful or harmful to others. In the event of cancellation or suspension, you will no longer be authorized to use or reproduce these datasets, and the City may use any means possible to enforce its decision. Such cancellation or suspension will not affect any person who has received the datasets from you and who is otherwise in compliance with these Terms of Use.

No Endorsement
You may not publicly represent or imply that the City is participating in, or has sponsored, approved, or endorsed the manner or purpose of, your use or reproduction of these datasets.

No Association
You may not use any trade-mark, official mark, official emblem or logo, of the City, or any of its other references or means of promotion or publicity without the City's prior written consent nor in any event to represent or imply an association or affiliation with the City.

Governing Law and Jurisdiction From Which DataSets Are Published
These datasets are published from within the Province of British Columbia, Canada. These Terms of Use are governed by British Columbia law and the City and you now irrevocably submit to the exclusive jurisdiction of British Columbia courts with respect to any and all matters arising under these Terms of Use or these datasets.

Annexe B à utiliser pour Exercice C3

```
#include <iostream>
#include <list>
#include <vector>

#include <time.h>

// ////////////////////////////////////////////////////////////////
// Team
// ////////////////////////////////////////////////////////////////

class Team
{
public:
     Team(void);
     Team(const std::string& name);
     Team(const Team& other);
    ~Team(void);

public:
    Team& operator=(const Team& other);

public:
    bool operator==(const Team& other);

public:
    operator bool(void) const;

public:
    friend std::ostream& operator<<(std::ostream& out, const Team&
team);

private:
    std::string m_name;
};

Team::Team(void)
{

}

Team::Team(const std::string& name) : m_name(name)
{

}

Team::Team(const Team& other) : m_name(other.m_name)
{
```

```
}

Team::~Team(void)
{

}

Team& Team::operator=(const Team& other)
{
    m_name = other.m_name;

    return (*this);
}

bool Team::operator==(const Team& other)
{
    return m_name == other.m_name;
}

Team::operator bool(void) const
{
    return m_name.length();
}

std::ostream& operator<<(std::ostream& out, const Team& team)
{
    out << team.m_name;

    return out;
}

// ////////////////////////////////////////////////////////////////
// Subject
// ////////////////////////////////////////////////////////////////

class Subject {};

// ////////////////////////////////////////////////////////////////
// X
// ////////////////////////////////////////////////////////////////

class X {};

// ////////////////////////////////////////////////////////////////
// Game
// ////////////////////////////////////////////////////////////////

class Game : public Subject, public X
{
public:
```

```
     Game(void);
     Game(const Game& other);
    ~Game(void);

public:
    Game& operator=(const Game& other);

public:
    void  addTeam(const Team& team);
    void finalize(const Team& team_a, int score_a, const Team& tema_b,
int score_b);

public:
    void attach(Game *game);
    void notify(void);

public:
    bool valid(const Team& team);

public:
    const Team& first(void) const;
    const Team& second(void) const;
    const Team& winner(void) const;

public:
    friend std::ostream& operator<<(std::ostream& out, const Game&
game);

private:
    std::list<Team> m_teams;

private:
    Team m_winner;

private:
    Game *m_next;
};

// ////////////////////////////////////////////////////////////////
// Tournament
// ////////////////////////////////////////////////////////////////

class Tournament
{
public:
     Tournament(void);
    ~Tournament(void);

public:
    void run(void);
```

```
public:
    friend std::ostream& operator<<(std::ostream& out, const
Tournament& tournament);

private:
    std::vector<Game> m_games;
    std::vector<Team> m_teams;
};

Tournament::Tournament(void)
{
    m_teams.push_back(Team("Team_1"));
    m_teams.push_back(Team("Team_2"));
    m_teams.push_back(Team("Team_3"));
    m_teams.push_back(Team("Team_4"));
    m_teams.push_back(Team("Team_5"));
    m_teams.push_back(Team("Team_6"));
    m_teams.push_back(Team("Team_7"));
    m_teams.push_back(Team("Team_8"));

    for(int i = 0; i < m_teams.size()/2; i++) {

        Game game;
        game.addTeam(m_teams[i]);
        game.addTeam(m_teams[m_teams.size()-1-i]);

        m_games.push_back(game);
    }
}

Tournament::~Tournament(void)
{

}

void Tournament::run(void)
{
    int n = m_games.size();
    int c = 0;

    srand(time(NULL));

    do {

        for(int i = c ; i < c+((n-c)/2); i++) {

            Game *next = new Game;

            Game *a = &(m_games[    i]);
```

```
            Game *b = &(m_games[n-1-i+c]);

            a->attach(next);
            b->attach(next);

            a->finalize(a->first(), rand()%100, a->second(),
rand()%100);
            b->finalize(b->first(), rand()%100, b->second(),
rand()%100);

            std::cout << (*a) << std::endl;
            std::cout << (*b) << std::endl;

            m_games.push_back(*next);
        }

        c = n;
        n = m_games.size();

        std::cout << std::endl;

    } while(c < n);

    Game *final = &(m_games.back());

    final->finalize(final->first(), rand()%100, final->second(),
rand()%100);

    std::cout << (*final) << std::endl;
    std::cout << std::endl;
    std::cout << "Tournament winner: " << final->winner() << std::endl;
}

std::ostream& operator<<(std::ostream& out, const Tournament&
tournament)
{
    for(int i = 0; i < tournament.m_games.size(); i++)
        out << tournament.m_games[i] << std::endl;

    return out;
}
int main(int argc, char **argv)
{
    Tournament tournament;
    tournament.run();

    return 0;
}
```