ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
Cours d'Information, Calcul et Communication, sections MA et PH
Chappelier J.-C.

Information, Calcul et Communication (SMA/SPH) : Correction de l'Examen III

20 décembre 2019

INSTRUCTIONS (à lire attentivement)

IMPORTANT ! Veuillez suivre les instructions suivantes à la lettre sous peine de voir votre examen annulé dans le cas contraire.

1. Vous disposez de deux heures quarante-cinq minutes pour faire cet examen (13h15 – 16h00).
2. Vous devez **écrire à l'encre noire ou bleu foncée**, pas de crayon ni d'autre couleur.
   N'utilisez **pas non plus de stylo effaçable** (perte de l'information à la chaleur).
3. Vous avez droit à toute documentation papier.
   En revanche, vous ne pouvez pas utiliser d'ordinateur personnel, ni de téléphone portable, ni aucun autre matériel électronique.
4. Répondez aux questions directement sur la donnée, **MAIS** ne mélangez pas les réponses de différentes questions !
   Ne joignez aucune feuilles supplémentaires ; **seul ce document sera corrigé**.
5. Lisez attentivement et *complètement* les questions de façon à ne faire que ce qui vous est demandé. Si l'énoncé ne vous paraît pas clair, ou si vous avez un doute, demandez des précisions à l'un des assistants.
6. L'examen comporte 6 exercices indépendants sur 12 pages, qui peuvent être traités dans n'importe quel ordre, mais qui ne rapportent pas la même chose (les points sont indiqués, le total est de 100 points) :
   1. questions courtes : 29 points ;
   2. routage IP : 11 points ;
   3. thés de Noël : 16 points ;
   4. programmation dynamique en C++ : 12 points ;
   5. virgule flottante : 13 points ;

   et
   6. atterrissage : 19 points.

   Tous les exercices comptent pour la note finale.

Question 1 – Varia [29 points]

Question 1.1 Caisse kiffée [6 points]

Considérez le programme suivant :

```
#include <iostream>
using namespace std;

void f1(int i, int j)
{
  int copie(i);
  i = j;
  j = copie;
}

int f2(int j)
{
  j = 1;
  return j-1;
}

int f3(int& k)
{
  k = 2;
  return k+1;
}
```

```
int f4(int* p)
{
  *p = 1;
  return *p;
}

int main()
{
  int i(1);
  int j(0);

  /* Ici on ne met que l'UNE de ces 4 lignes :
     f1(i,j);
     i = f2(j);
     j = f3(i);
     i = f4(&j);
  */

  cout << "i = "   << i
       << "; j = " << j << endl;

  return 0;
}
```

Pour chacun des appels envisagés (seul) dans le `main()` ci-dessus, dire ce que le programme affiche.

- avec uniquement l'appel à `f1`, le programme affiche : `i = 1 ; j = 0`
  il n'y a PAS de passage par référence ici !
- avec uniquement l'appel à `f2`, le programme affiche : `i = 0 ; j = 0`
- avec uniquement l'appel à `f3`, le programme affiche : `i = 2 ; j = 3`
- avec uniquement l'appel à `f4`, le programme affiche : `i = 1 ; j = 1`

Question 1.2 Future licorne ou arnaqueurs ? [9 points]

Une start-up vend un algorithme de compression de données textuelles (évidemment non ambigu !) et prétend comprimer à 80% (c.-à-d. 80% de gain) des textes constitués de lettres ayant une entropie de 2 bits, mais représentées au départ chacune sur 8 bits.

① **[3 points]** Qu'en pensez-vous ? Expliquez brièvement votre réponse dans la place impartie :

Réponse : C'est possible avec pertes, mais **pas sans perte** : 80% de gain signifie que la longueur moyenne par caractère est de $20\% \times 8$ bits, soit 1.6 bits, ce qui est plus petit que l'entropie ; ce qui est impossible car contradictoire avec le théorème de Shannon.

Cette même start-up prétend que son algorithme est sans perte, et publie des expériences le comparant aux algorithmes de Huffman et de Shannon-Fano.

Elle reporte quatre expériences en publiant à chaque fois (en bits) la longueur L_{SF} obtenue en utilisant l'algorithme de Shannon-Fano, la longueur L_{Hu} obtenue en utilisant l'algorithme de Huffman, la longueur L_{eux} obtenue en utilisant leur algorithme, et l'entropie H du texte utilisé.

② **[6 points]** Pour *chacune* des expériences suivantes, dites si cela vous semble crédible et expliquez à chaque fois brièvement pourquoi :

- $L_{\text{SF}} = 6.3$, $L_{\text{Hu}} = 5.1$, $L_{\text{eux}} = 5.1$ et $H = 4.5$:
  impossible car $L_{\text{SF}} \geq H + 1$
- $L_{\text{SF}} = 5.8$, $L_{\text{Hu}} = 4.5$, $L_{\text{eux}} = 5.5$ et $H = 5.1$:
  impossible car $L_{\text{Hu}} < H$ (th. de Shannon)
- $L_{\text{SF}} = 6.3$, $L_{\text{Hu}} = 6.2$, $L_{\text{eux}} = 6.1$ et $H = 6$:
  impossible car $L_{\text{eux}} < L_{\text{Hu}}$ qui est optimal (sans perte)
- $L_{\text{SF}} = 6.5$, $L_{\text{Hu}} = 6.4$, $L_{\text{eux}} = 6.4$ et $H = 6.2$:
  possible : $H \leq L_{\text{Hu}} \leq L_{\text{SF}} < H + 1$ et $L_{\text{Hu}} \leq L_{\text{eux}}$
  (mais pas forcément très intéressant : pourquoi ne pas simplement utiliser le code de Huffman ?)

Question 1.3 Filtres [6 points]

BAREME : 1 point par question.

On considère le signal $X(t)$ suivant :

$$X(t) = 5\,\sin(7\,\pi\,t + \frac{\pi}{6}) + 6\,\sin(6\,\pi\,t + \frac{\pi}{12}) + 3\,\sin(8\,\pi\,t + \frac{\pi}{8})$$

① **[1.5 points]** On applique à ce signal $X(t)$ un filtre passe-bas idéal de fréquence de coupure $f_c = 3.75$ Hz. Quel signal $Y(t)$ obtient-on ?

$$Y(t) = 5\,\sin(7\,\pi\,t + \frac{\pi}{6}) + 6\,\sin(6\,\pi\,t + \frac{\pi}{12})$$

② **[1 point]** On s'intéresse au signal $Z(t) = X(t) - Y(t)$. Quelle est sa bande passante ?

$$B = 4 \text{ Hz}$$

③ **[1.5 points]** Si on échantillonne $Z(t)$ à une fréquence $f_e = 7$ Hz, peut-on assurer une reconstruction parfaite ? Justifiez *brièvement* votre réponse.

Non (th. de Nyquist-Shannon). Ici, il n'y a pas d'autre solution que d'augmenter la fréquence d'échantillonnage au dessus de deux fois la bande passante, soit 8 Hz (car filtrer enlèverait tout le signal (1 seule fréquence)).

④ **[2 points]** Finalement, on décide d'échantillonner $Z(t)$ à une fréquence $f_e = 9$ Hz. Quel est le signal $\widehat{Z}(t)$ obtenu après reconstruction ?

$$\widehat{Z}(t) = Z(t)$$

Note : la formule explicite de $Z(t)$ (à savoir : $3\,\sin(8\,\pi\,t + \frac{\pi}{8})$) est bien sûr une réponse correcte.

Question 1.4 La matrice cachée [8 points]

Sur un ordinateur avec de la mémoire cache, on appelle la fonction `applique()` correspondant au code C++ suivant :

```
1   typedef vector<double> Vecteur;
2   typedef vector<Vecteur> Matrice;
3
4   Vecteur applique(Matrice const& m, Vecteur const& v)
5   {
6     Vecteur retour(v);
7     for (size_t i(0); i < m.size(); ++i) {
8       retour[i] = 0.0;
9       for (size_t j(0); j < m[i].size(); ++j) {
10          retour[i] += m[i][j] * v[j];
11      }
12    }
13    return retour;
14  }
```

① **[2 points]** Le code ci-contre est-il correct ? Sinon, expliquez pourquoi.

Il est globalement correct, **mais** les accès à `retour` et à `v` ne sont pas protégés : il n'est pas dit que `v` ait la bonne taille....

② **[6 points]** Si l'on nomme « *ligne* » la première dimension d'indexation de `m` (donc `m[i]` est une « ligne de `m` »), dites pour chacune des propositions suivantes si elle est vraie ou fausse pour le *premier* appel à `applique()` (éventuellement corrigée) dans le programme.
Justifiez à chaque fois *brièvement* votre réponse.

A] Si la cache est assez grande pour contenir en même temps une `Matrice` et deux `Vecteur`, alors il n'y aura aucun défaut de cache.

FAUX, car il y aura les défauts de cache au démarrage (premier appel).

B] Si la cache est assez grande pour contenir trois `Vecteur`, alors le nombre de défauts de cache ne dépend pas de la façon dont la `Matrice m` est stockée en mémoire.

VRAI ici (C++) car une ligne de `m` est un `Vecteur` (mais serait faux en général car on ne saurait pas assez de choses sur la façon dont la `Matrice m` est stockée en mémoire).

C] Si les accès mémoire et la cache sont organisés de sorte à ce que la cache puisse contenir en même temps deux `Vecteur` et tous les `m[i][j]` pour `j` fixé et tous les `i` (colonne de `m`), alors le nombre de défauts de cache est le même que si la cache était assez grande pour contenir en même temps une `Matrice` et deux `Vecteur`.

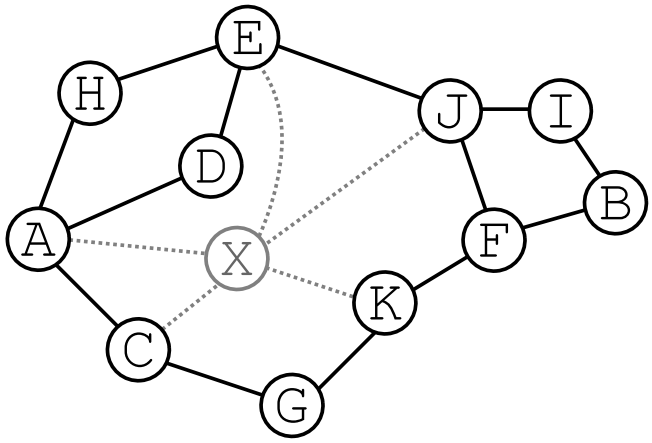
FAUX : contenir une colonne en cache n'apporte rien ici vu que la boucle interne se fait sur une ligne donnée.

D] Si les accès mémoire et la cache sont organisés de sorte à ce que la cache puisse contenir en même temps deux `Vecteur` et tous les `m[i][j]` pour `i` fixé et tous les `j` (ligne de `m`), alors le nombre de défauts de cache est le même que si la cache était assez grande pour contenir en même temps une `Matrice` et deux `Vecteur`.

VRAI puisque justement la boucle intérieure se fait sur une ligne (et c'est toujours vrai, C++ ou non).

Question 2 Routage IP [11 points]

On considère le réseau de routeurs suivant :



Pour simplifier, on se limitera dans tout cet exercice aux tables de routage **à distance 2 maximum**. Par exemple, le nœud `B` n'indiquera rien sur le nœud `G`.

Question 2.1 Tables [3 points]

① **[1.5 points]** Quelle est la table de routage du routeur `K`, *si le nœud* ***`X` n'est pas*** *présent* :

dest.	dist.	route
G	1	G
F	1	F
C	2	G
J	2	F
B	2	F

② **[1.5 points]** Quelle est la table de routage du routeur `K`, *si nœud* ***`X` est*** *présent* :

dest.	dist.	route
X	1	X
A	2	X
E	2	X
G	1	G
F	1	F
C	2	G ou X, mais pas les deux
J	2	F ou X, mais pas les deux
B	2	F

Question 2.2 Modification des tables [3 points]

Chaque nœud communique à ses voisins les changements de sa table de routage.

③ **[1 point]** Que communique le nœud `K` à ses voisins lorsque le nœud `X` est introduit ?

C'est simplement la différence entre les 2 tables précédentes :

X	1	(x)
A	2	(x)
E	2	(x)

+ optionnels, s'ils ont été changés dans la réponse précédente :

C	2	(x)
J	2	(x)

(Note : la troisième colonne n'est en toute rigueur par communiquée.)

④ **[2 points]** Que communique le nœud `J` à ses voisins lorsque le nœud `X` est introduit ?

X 1 (x)
A 2 (x)
C 2 (x)

Question 2.3 Rôle des routeurs [2 points]

A quoi sert d'introduire un tel nœud `X` dans un réseau ?

Cela ajoute de la **redondance** de route pour rendre le réseau plus résistant aux pannes de nœuds.

Par ailleurs, cela permet de réduire les distances (la **latence**).

Question 2.4 Complexité [3 points]

A quelle classe de complexité appartient le problème qui consiste à trouver une route pour des paquets entre deux ordinateurs dans un réseau IP ?

Justifiez votre réponse.

C'est forcément dans P, sinon Internet ne fonctionnerait simplement pas !!
Notez que nous parlons bien ici du *problème*, pas de tel ou tel algorithme utilisé concrètement en pratique.

Autre argument : c'est un problème de plus court chemin dont on connaît au moins une solution polynomiale (même si ce n'est pas forcément cette solution qui est finalement implémentée).

Troisième argument : la complexité de l'algorithme du calcul des tables de routages vu en classe est au pire cubique (au pire quadratique pour chaque routeur).

Question 3 Thés de Noël [16 points]

Question 3.1 Quel bazar ! [6 points]

Voulant vous faire un thé, vous avez malheureusement renversé la boîte qui contenait tous les sachets et les voilà tous mélangés. Il y avait 22 sachets en tout : 10 de « breakfast tea », 5 de thé de Noël, 2 de thé vert, 2 de thé au jasmin, 2 de rooïbos aux épices et 1 de thé Chai.

Quelle est, en bit, l'entropie du choix d'un sachet de thé au hasard ?

Donnez votre réponse sous la forme $a + b\,\log_2(5) + c\,\log_2(11)$ en exprimant la valeur de a, de b et de c sous forme de fraction irréductible :

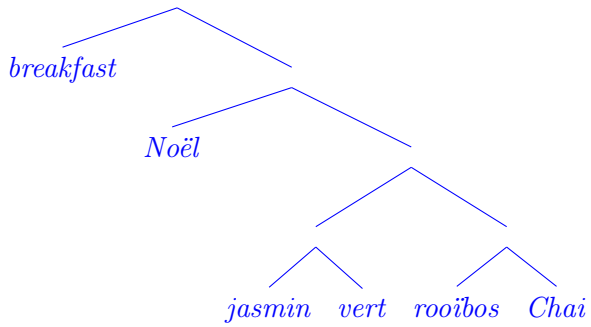
$$a = \frac{3}{11} \qquad b = -\frac{15}{22} \qquad c = 1$$

Explication (non demandée) :

$$\begin{aligned}&\frac{10}{22}\log_2\frac{22}{10} + \frac{5}{22}\log_2\frac{22}{5} + \frac{6}{22}\log_2\frac{22}{2} + \frac{1}{22}\log_2 22\\ =&\log_2 22 - \frac{10}{22}\log_2 2 - \frac{10}{22}\log_2 5 - \frac{5}{22}\log_2 5 - \frac{6}{22}\log_2 2\end{aligned}$$

Question 3.2 Encodez moi tout ça ! [4 points]

Proposez un code sans-préfixe optimal pour représenter les différents thés (correspondants à la situation ci-dessus). Donnez le nom du code que vous construisez, puis faites le.



Ce qui donne les longueurs suivantes :

- *breakfast* : 1
- *Noël* : 2
- *jasmin* : 4
- *vert* : 4
- *rooïbos* : 4
- *Chai* : 4

Question 3.3 Cohérence ? [6 points]

Quelle est la longueur moyenne de votre code ? Est-ce cohérent, sachant que $3\log_2(5) \simeq 7$ et $\log_2(11) \simeq \frac{76}{22}$? Justifiez *pleinement* votre réponse.

$$\overline{L} = \frac{10}{22}\times 1 + \frac{5}{22}\times 2 + \frac{7}{22}\times 4 = \frac{48}{22} = \frac{24}{11}$$

Avec les approximations données, on trouve que l'entropie est environ égale à $\frac{47}{22}$.

Ces résultats sont cohérents (bien sûr !) avec le théorème de Shannon (partie universelle) : la longueur moyenne est supérieure ou égale à l'entropie.

(On pourrait aussi mentionner que cette longueur moyenne est inférieure à l'entropie plus 1 bit ; ce qui est cohérent avec la combinaison de l'optimalité du code de Huffman et la majoration (ad-hoc) du code de Shannon-Fano.)

Question 4 Programmation dynamique en C++ [12 points]

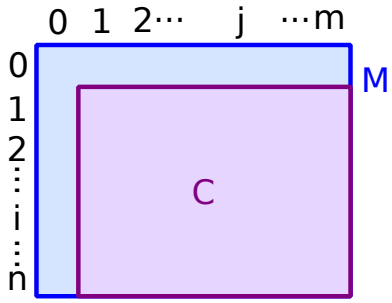
Soit C une matrice réelle de taille $n \times m$. On veut calculer les valeurs d'une matrice réelle M de taille $(n+1) \times (m+1)$ telle que :

- $M(0, j) = 0$ pour tout j de 0 à m ;
- $M(i, 0) = 0$ pour tout i de 1 à n ;
- $M(i, j) = C(i, j) + \max\big(M(i, j-1), M(i-1, j), M(i-1, j-1)\big)$ pour tous les autres indices $(1 \leq i \leq n$ et $1 \leq j \leq m)$.

① **[10 points]** Ecrire **en C++** une fonction `align()` permettant de calculer M à partir de C.
Si vous utilisez vos propres types, définissez-les avant la fonction.
Si C n'est pas conforme, lancez une exception.

② **[2 points]** Quelle est la complexité de l'algorithme correspondant à votre fonction `align()` ?

La principale difficulté de cet exercice réside dans la gestion des indices des matrices (pour information, il s'agit d'un algorithme d'alignement de séquences par programmation dynamique) :



[Pour les problèmes de gestion d'indices entre mathématiques et C++, voir p.ex. l'exercice 6.2 de la série 7 (« tri de Shell »).]

Ensuite, notez que pour pouvoir calculer $M(i, j)$, il est nécessaire que les cases $M(i, j-1)$, $M(i-1, j)$ et $M(i-1, j-1)$ aient été calculées au préalable. Il y a pour cela *plusieurs* parcours possibles, en voici *un possible* (par lignes ; on peut aussi procéder par colonnes ou même par anti-diagonales) :

```
typedef vector<vector<double>> Matrice;

Matrice align(Matrice const& C)
{
  if (C.empty()) return Matrice(1, vector<double>(1));

  Matrice M(C.size() + 1, vector<double>(C[0].size() + 1));

  for (size_t i(1); i < M.size(); ++i) {
    if (C[i-1].size() != M[i].size() - 1) throw "erreur"; // whatever

    for (size_t j(1); j < M[i].size(); ++j) {
      M[i][j] = C[i-1][j-1] + max(max(M[i][j-1], M[i-1][j]), M[i-1][j-1]);
    }
  }

  return M;
}
```

Au niveau du type, on peut aussi ici *tolérer* des `array`, si l'on fait l'hypothèse que n et m sont connus et fixés. Ceci a par contre l'avantage de ne pas avoir à tester la taille de chaque `C[i-1]`, mais a l'inconvénient de devoir définir *deux* types : un pour C et un pour M (car elles sont de tailles différentes).

L'algorithme ci-dessus est en $\mathcal{O}(t)$ avec $t = n \times m$ la taille de la matrice C en entrée.

Question 5 Virgule flottante [13 points]

On s'intéresse ici à créer notre propre représentation en virgule flottante en C++.

Pour simplifier, nous ne nous intéressons dans cette question qu'à des nombres de valeur absolue supérieure ou égale à 1 et nous fixons le nombre de bits de la mantisse à 5 et le nombre de bits de l'exposant à 3.

Question 5.1 Exemple [5 points]

À quelle valeur correspond la séquence de bits `111010110`, sachant qu'elle est donnée dans l'ordre signe, mantisse, exposant ?

Justifiez votre réponse.

`111010110` se lit comme `1 11010 110`, soit un nombre négatif, de mantisse égale à `1.11010` (en binaire, soit $1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{16}$), et d'exposant 6.

Le nombre représenté est donc $-(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{16}) \times 2^6 = -(64 + 32 + 16 + 4) = -116$.

Question 5.2 Affichage en C++ [8 points]

On suppose, toujours pour simplifier, représenter de tels nombres en C++ à l'aide de la structure suivante :

```
struct virgule_flottante {
  int signe;
  int mantisse;
  int exposant;
};
```

Définissez sur la page ci-contre une fonction `affiche()` permettant d'afficher la valeur (décimale, à virgule) correspondant au contenu d'une telle structure (toujours en supposant 5 bits utilisés pour la mantisse et 3 bits pour l'exposant).

L'affichage devra comporter trois parties (exemple ci-dessous) : la valeur décimale signée (et à virgule) correspondant à la mantisse (`1.1875` ci-dessous), la valeur de l'exposant (`2` ci-dessous) et la valeur totale (`4.75` ci-dessous). Par exemple, l'appel `affiche({ 0, 6, 2 });` produira l'affichage suivant :

```
1.1875 * 2^2 = 4.75
```

Notes : on pourra faire l'hypothèse que tous les nombres représentés par `virgule_flottante` ont une valeur correspondante qui est représentable sur un `double`.

Il y a plein de solutions possibles. En voici deux :

```
void affiche(virgule_flotante const& x)
{
  double y(1.0);
  y += x.mantisse / 32.0;
  if (x.signe == 1) {
    y = -y;
  }
  cout << y << " * 2^" << x.exposant;
  y *= pow(2.0, x.exposant);
  cout << " = " << y << endl;
}
```

```
void affiche(virgule_flotante const& x)
{
  if (x.signe == 1) {
    cout << '-';
  } else {
    cout << '+';
  }

  int decim(x.mantisse);
  int count(5);
  double y(1.0);
  while (decim != 0) {
    if (decim % 2) y += pow(2.0, -count);
    decim /= 2;
    --count;
  }

  cout << y << " * 2^" << x.exposant;

  y *= pow(2.0, x.exposant);
  if (x.signe == 1) y = -y;
  cout << " = " << y << endl;
}
```

Notes :

- le `32.0` de la solution de gauche ou le `5` de la solution de droite seraient bien sûr définis comme des `constexpr` dans un programme plus large ;
- le code (inutile) de la conversion `int` en `double` de la mantisse au milieu de la solution de droite mériterait, bien sûr, d'être modulariser (c.-à-d. en faire une fonction).

Question 6 Atterrissage [19 points]

On vous demande d'écrire sur la page ci-contre un programme C++ (simple) pour jouer au « jeu de l'atterrissage » : faire atterrir en douceur une fusée ayant une quantité de carburant limitée.

La fusée descend en chute libre, sauf si le moteur est allumé pour ralentir son mouvement. Si le moteur est allumé trop peu ou trop tard, elle s'écrasera ; par contre, si le moteur est allumé trop tôt ou trop fort, on risque la panne sèche et donc finalement de s'écraser par manque de carburant.

Voici un exemple de déroulement possible :

```
hauteur : 500, vitesse : -50, carburant restant : 120
quantite a consommer : -10
--> quantité invalide
quantite a consommer : 150
--> quantité invalide (plus assez de carburant)
quantite a consommer : 10
hauteur : 452.5, vitesse : -45, carburant restant : 110
quantite a consommer : 0
hauteur : 405, vitesse : -50, carburant restant : 110
...
hauteur : 45, vitesse : -40, carburant restant : 50
quantite a consommer : 20
hauteur : 12.5, vitesse : -25, carburant restant : 30
quantite a consommer : 25.5
Atterri en douceur.
```

Le programme doit avoir le comportement suivant (voir exemple de déroulement ci-dessus) : tant que la fusée n'est pas arrivée (c.-à-d. que sa hauteur est positive), il affiche les informations de hauteur, vitesse et quantité de carburant restant, puis demande au joueur la quantité q de carburant à injecter dans le moteur. Celle-ci ne peut en aucun cas être négative, ni supérieure à la quantité de carburant restant.

On suppose que l'accélération totale, a, de la fusée est de la forme $a = q - q_0$, où q_0 est la quantité de carburant à injecter dans le moteur pour compenser la force de gravitation (c'est une constante connue ; disons 5).

Lorsque la fusée est arrivée (hauteur nulle ou négative), le programme affiche « *Atterri en douceur.* » si la hauteur est supérieure à $-q_0/2$ *et* la vitesse a une valeur absolue inférieure à q_0 ; sinon, il affiche « *Crash !* ».

Les formules de mise à jour de la position et de la vitesse de la fusée sont les suivantes (dans cet ordre, c.-à-d. mise à jour de v *après* h) :

$$\Delta h = v + \frac{1}{2}\,a\,, \qquad\qquad \Delta v = a\,,$$

où h est la hauteur de la fusée, v sa vitesse et Δx représente la variation de x à chaque itération.

On mettra explicitement dans le code les valeurs initiales suivantes : $h = 500$, $v = -50$ et la quantité initiale de carburant sera de 120.

Indication : n'oubliez pas de mettre à jour le carburant.

```cpp
#include <iostream>
#include <cmath> // abs()
using namespace std;

constexpr double q0(5.0);

double demander_q(double max)
{
  double q(0.0);
  do {
    cout << "quantité à consommer : ";
    cin >> q;
    if (q < 0.0) cout << "--> quantité invalide" << endl;
    if (q > max) cout << "--> quantité invalide (plus assez de carburant)" << endl;
  } while ((q < 0.0) or (q > max));
  return q;
}

int main()
{
  double h(500.0);
  double v(-50.0);
  double r(120.0);

  constexpr double h_crash(-0.5*q0);
  constexpr double v_crash(q0);

  while (h > 0) {
    cout << "hauteur : " << h
         << ", vitesse : " << v
         << ", carburant restant : " << r
         << endl;

    const double q = demander_q(r);
    const double a = q - q0;

    h += v + 0.5 * a;
    v += a;
    r -= q;
  }

  if ((h > h_crash) and (abs(v) < v_crash)) {
  // ces inégalités peuvent être larges (non précisé)

    cout << "Atterri en douceur." << endl;
  } else {
    cout << "Crash !!" << endl;
  }

  return 0;
}
```