TEST2 Sem1 MT & EL 22.11.2019 EPFL / R. Boulic

W=blanc, B=bleu, S=saumon, Y=jaune, All = même réponse pour tous

1) (3 pts) Compléter la fonction singleNumber()

Détection de la valeur qui n'apparaît qu'une seule fois dans un vector
La fonction **singleNumber** reçoit en paramètre un vector non-trié contenant au moins **3** éléments. Nous savons aussi qu'il contient *une valeur qui n'apparait qu'une seule fois* tandis que les autres valeurs apparaissent en double. Les valeurs en double ne sont pas forcément consécutives dans le vector.

La fonction **singleNumber** doit renvoyer la valeur qui n'apparaît qu'une seule fois. Par exemple, si on fournit un vector contenant ces valeurs **{5, 3, 1, 2, 2, 5, 1}** alors la fonction renvoie **3** car c'est la seule valeur qui n'apparaît qu'une seule fois.

```
1  // on suppose que les #include nécessaires sont faits
2  int singleNumber(vector<int>& nums)
3  {
4      // sorting the numbers from minimum to maximum
5      sort(nums.begin(), nums.end());
6
7      for (int i = 0; i < nums.size()-1; __ ① ___ )
8      {
9          if (nums[i] != nums[i + 1])
10         {
11             return ___ ② ___;
12         }
13     }
14
15     return nums[nums.size()-1];
16 }
```

Choisir parmi une des réponses proposées en justifiant votre réponse par deux à trois phrases ; n'hésitez pas à faire des dessins comme support à votre explication :

➔ A : ① : **i = i + 1** et ② : **nums[i]**

➔ B : ① : **i = i + 1** et ② : **nums[i+1]**

➔ C : ① : **i = i + 2** et ② : **nums[i] // [W]**

➔ D: ① : **i = i + 2** et ② : **nums[i+1]**

A for [B], D for [S], B for [Y]

Réponse choisie, avec sa JUSTIFICATION :[All]
L'algorithme sort() (vu dans la série9) permet de trier le vector nums dans l'ordre croissant. Les valeurs présentent en double sont donc consécutives après le tri. Pour permettre de trouver la valeur isolée, le test de la ligne 9 doit être fait au sein de chaque paire de valeurs : donc il faut parcourir le vector par pas de **2** pour passer d'une paire à la suivante. La valeur isolée est celle de l'indice **i**.

2) (4 pts) Fonction récursive f ()

Le programme suivant donne un résultat correct quand la fonction récursive **f ()** est appelée avec des valeurs v1 et v2 appartenant à l'intervalle **] 0, 2^{16}[** .

```
1   #include <iostream>
2   using namespace std;
3
4   unsigned int f(unsigned int v1, unsigned int v2);
5
6   int main ()
7   {
8        unsigned int v1, v2;
9
10       do
11       {
12            cin >> v1 >> v2 ;
13       }while(v1==0 or v2==0);
14
15       cout << f(v1, v2);
16
17       return 0;
18  }
19
20  unsigned int f(unsigned int v1, unsigned int v2)
21  {
22       if(v2 > 1)
23            return f(2*v1, v2/2) + ((v2%2)? v1 : 0);
24
25       return v1;
26  }
```

2.1) **Indiquer la valeur renvoyée par f ()** pour les appels suivants en détaillant les appels récursifs (valeur des paramètres, valeur renvoyée) s'il y en a.

f (1, 1) renvoie : 1 [All] 1 >1 est faux donc renvoie v1 qui vaut 1	**f (1, 2) renvoie : 2 [All]** 2>1 return f(2,1) + 0 f(2,1) 1>1 est faux donc renvoie 2 donc f(1,2) renvoie 2+ 0 = 2
f (2, 3) renvoie : 6 [W] 3>1 return f(4,1) + 2 f(4,1) 1>1 est faux donc renvoie 4 donc f(2,3) renvoie 4 + 2 = 6	**f (7, 3) renvoie : 21 [W]** 3>1 return f(14,1) + 7 f(14,1) 1>1 est faux donc renvoie 14 donc f(7,3) renvoie 14 + 7 = 21
f (3,2) renvoie : 6 [B] 2>1 return f(6,1) + 0 f(6,1) 1>1 est faux donc renvoie 6 donc f(3,2) renvoie 6 + 0 = 6	**f (5,4) renvoie : 20 [B]** 4>1 return f(10,2) + 0 f(10,2) return f(20,1) + 0 f(20,1) 1>1 est faux donc renvoie 20 donc f(2,3) renvoie (20 + 0) +0 = 20

f (2, 3) renvoie : 6 [S] 3>1 return f(4,1) + 2 f(4,1) 1>1 est faux donc renvoie 4 donc f(2,3) renvoie 4 + 2 = 6	**f (6, 3) renvoie : 18 [S]** 3>1 return f(12,1) + 6 f(12,1) 1>1 est faux donc renvoie 12 donc f(6,3) renvoie 12 + 6 = 18
f (3,2) renvoie : 6 [Y] 2>1 return f(6,1) + 0 f(6,1) 1>1 est faux donc renvoie 6 donc f(2,3) renvoie 6 + 0 = 6	**f (4,5) renvoie : 20 [Y]** 5>1 return f(8,2) + 4 f(8,2) return f(16,1) + 0 f(16,1) 1>1 est faux donc renvoie 16 donc f(4,5) renvoie (16 +0) + 4 = 20

2.2) **Quel est le but de la fonction f () ? [All]**

Il s'agit de l'algorithme de la multiplication égyptienne vu dans la partie théorique module1, ici dans une version récursive. Il s'appuie sur un algorithme de conversion décimal->binaire.

2.3) **Maintenant supposons que v1 et v2 ont pour valeur 2^{16}** qui est juste en dehors du domaine autorisé pour les paramètres de f(). En déduire (sans détailler l'appel de f()) la valeur renvoyée par f() ; pourquoi obtient-on ce résultat ?

On obtient un motif binaire nul, c'est-à-dire 0.

En effet le produit de 2^{16} par 2^{16} donne 2^{32} qui en théorie a un poids de 1 pour 2^{32} et 0 pour tous les poids des puissances inférieures de 2.

Or 2^{32} n'est pas représentable en tant qu'entier **int** car un tel entier ne dispose que de 32 bits et que la plus grande puissance est 2^{31}.

W=blanc, B=bleu, S=saumon, Y=jaune, All = même réponse pour tous

3) (4 pts) String : que fait la fonction g ()

Le programme suivant donne un résultat correct.

```
1   #include <iostream>
2   #include <vector>
3   #include <algorithm>
4   using namespace std;
5
6   string g(vector<string> &t)
7   {
8       if(t.size() == 0)
9           return "";
10      else if(t.size() == 1)
11          return t[0];
12
13      size_t len = t[0].size();
14      for(size_t i = 1; i<t.size();i++)
15      {
16          len = min(len, t[i].size());
17      }
18
19      string str = "";
20      size_t j = 0;
21      while(j < len)
22      {
23          size_t i = 0;
24          while((i < t.size()-1) && (t[i][j] == t[i+1][j]))
25              i++;
26
27          if(i == t.size() -1)
28              str += t[0][j];
29          else
30              break;
31
32          j++;
33      }
34      return  str;
35  }
36
37  int main()
38  {
39      vector<string> tab = {"monday","monkey","monster"};
40      cout << g(tab) << endl;
41      return 0;
42  }
```

Remarque : #include <string> est obtenu avec #include <iostream>

3.1) Quel est l'affichage obtenu à la ligne 40 avec cout :

mon [W], mor [B], mod [S], mos [Y]

3.2) Justifier votre réponse du 3.1) en indiquant comment la fonction **g()** traite le paramètre formel **t** pour obtenir ce résultat. Indiquer les résultats intermédiaires s'il y en a. Conclure en indiquant le but de cette fonction : [All]

Le paramètre t est un vector de string. Le début de g() vérifie la taille de t pour traiter 2 cas simples : si t est vide g() renvoie la chaine vide ; si t est réduit à un seul élément g() renvoie cet élément. Cela n'est pas suffisant pour déterminer le but de g().

Le reste de g() traite donc un ensemble d'au moins 2 string. Les lignes 13-17 détermine la plus petite longueur des string de t, noté **len**.

Ensuite la boucle while ligne 20 examine le **j**$^{\text{ième}}$ caractère de toutes les strings mais au maximum les **len** premiers caractères. Cette boucle construit la chaîne **str** initialement vide en ajoutant un caractère à chaque passage dans la boucle while.

La condition à remplir est que le caractère d'indice **j** est présent dans toutes les strings de **t** qui sont parcourues avec la boucle ligne 24. Cela est vérifié ligne 27 quand i est égal à taille-de-t - 1.

On quitte la boucle sur j dès le premier caractère différent. Cette fonction renvoie donc la **sous-chaine de caractères commune au début de l'ensemble des chaînes du vector t.**

C'est bien le cas avec la sous-chaine indiquée en solution de 3.1)

3.3) Que se passe-t-il si on remplace la boucle des lignes 24-25 par celles-ci :

```
while(i < t.size()-1)
    if(t[i][j] == t[i+1][j])
        i++;
```

Préciser si une erreur est signalée par le compilateur ou pas. Si oui laquelle.
Si non que se passe-t-il à l'exécution ? [All]

Il n'y a pas d'erreur de compilation.

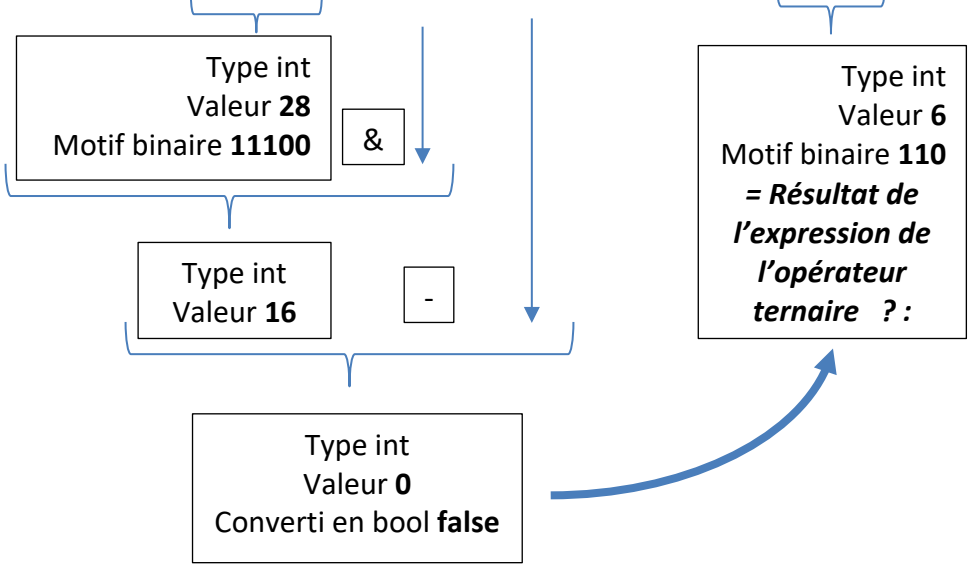
Il y a par contre une erreur sémantique car l'exécution produit une boucle infinie dès le premier caractère différent entre deux strings du vector t. En effet la variable i n'est plus incrémentée et on continue à tester deux élément différents sans fin.

W=blanc, B=bleu, S=saumon, Y=jaune, All = même réponse pour tous

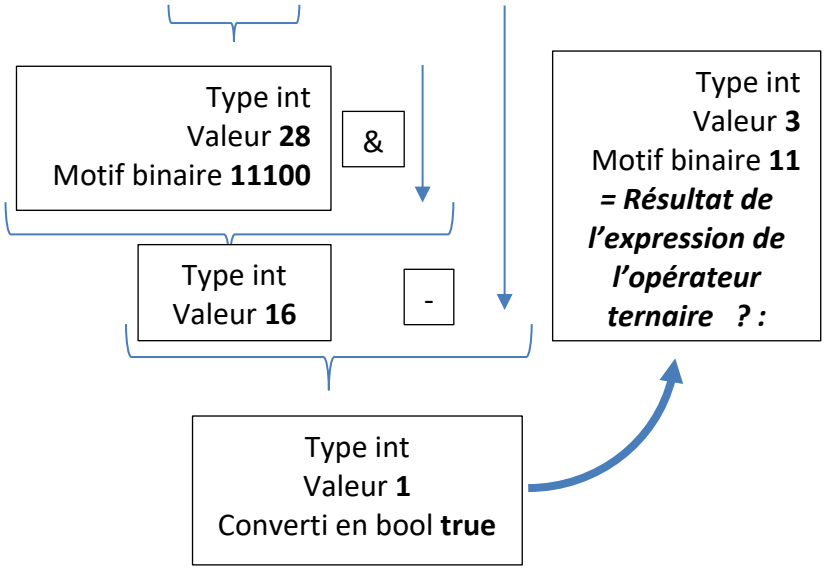
4) (2 pts) évaluation d'expression

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.

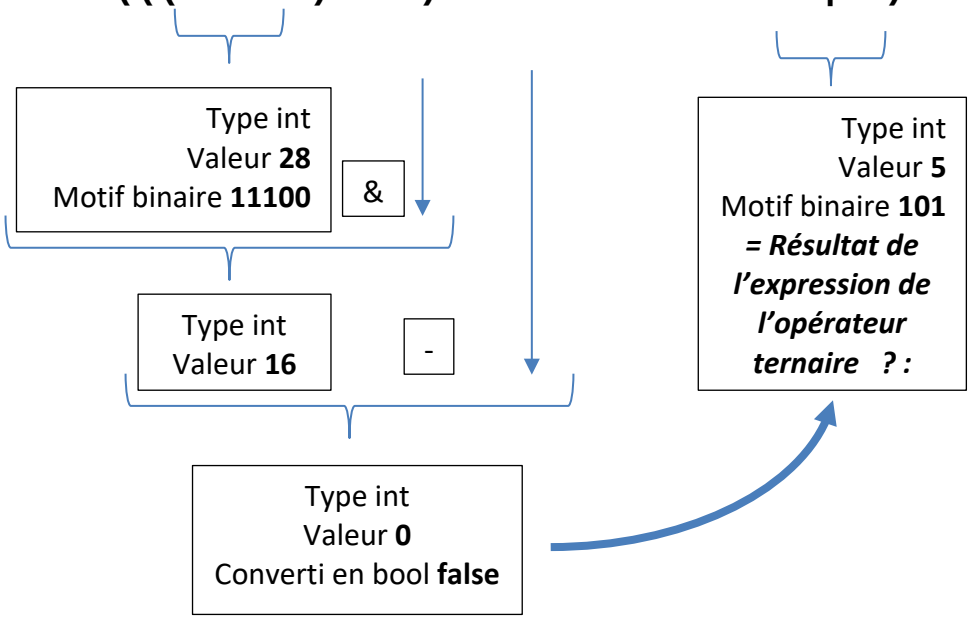
```
cout << (((7 << 2)& 48) - 16 ? 1 ^ 2 : 4 ¦ 2) << endl;[W]
```



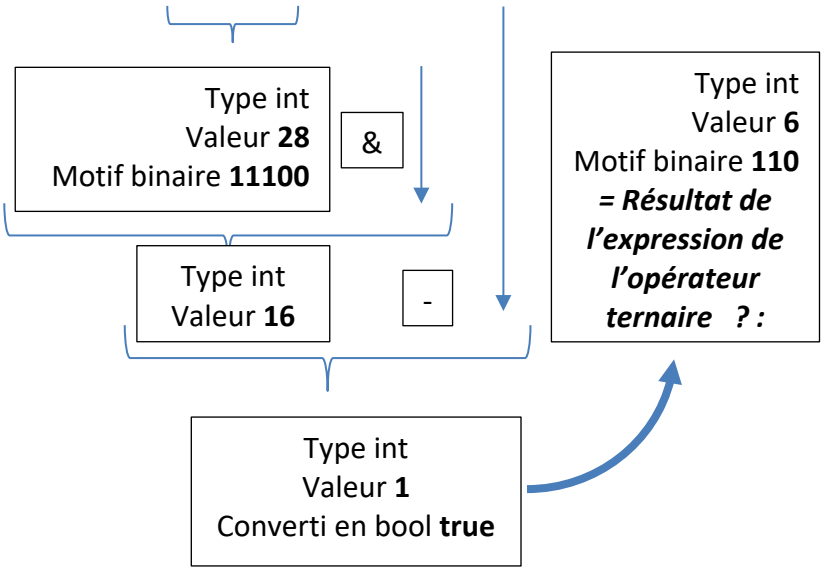

```
cout << (((7 << 2)& 48) - 15 ? 1 ^ 2 : 4 ¦ 2) << endl;[B]
```



```
cout << (((7 << 2)& 48) - 16 ? 1 ^ 2 : 4 ¦ 1) << endl;[S]
```




```
cout << (((7 << 2)& 48) - 15 ? 4 ^ 2 : 4 ¦ 1) << endl;[Y]
```



5) (6 pts) Application de gestion d'un hotel contenant une erreur sémantique:

Le programme suivant compile correctement avec C++11.

Ce programme de gestion d'un hotel doit permettre :

- **d'ajouter** un client dans une des 10 chambres numérotées de 0 à 9
(commande **Add** indiquée en fournissant la lettre **A**)

- de **libérer** une des chambres déjà occupées
(commande **Empty** indiquée en fournissant la lettre **E**)

- **d'afficher** la liste des noms de clients qui ont une chambre actuellement dans l'hotel
(commande **Display** indiquée en fournissant la lettre **D**).

Nous supposons qu'un nom de client est limité à un seul mot (ça n'est pas l'erreur recherchée).

5.1) Tout d'abord examinons un comportement *correct* du programme au niveau de la fonction **readRoomNumber.** Expliquer comment cette fonction s'exécute si on fournit la valeur **-1** au clavier quand elle lit la variable roomNumber : [All]

```
void readRoomNumber(unsigned int& roomNumber)
{
    do
    {
      cout << "Provide a Room Number in [0, 9]: " ;
      cin  >> roomNumber;
    }    while(roomNumber > 9);
}
```

La valeur -1 en complément à deux a un motif binaire avec 1 pour tous les bits.

Ce motif binaire correspond à la valeur positive la plus grande représentable avec le type unsigned int.

Cette valeur est supérieure à 9 en particulier, donc la condition du while est true, ce qui produit une nouvelle demande de lecture de roomNumber car la valeur correspondant à -1 est considérée comme incorrecte.

Cette fonction est donc correcte car elle garantit d'obtenir un indice compris entre 0 et 9 inclus.

```
1   #include <iostream>
2   #include <vector>
3   #include <string>
4   using namespace std;
5
6   struct Room
7   {
8       string client;
9       bool occupied;
10  };
11
12  void addClient(vector<string>& client, vector<Room>& hotel);
13  void emptyRoom(vector<string>& client, vector<Room>& hotel);
14  void displayClient(const vector<string>& client);
15  void readRoomNumber(unsigned int& roomNumber);
16
17  int main()
18  {
19      vector<string> client;
20      const unsigned int nb_room(10);
21      vector<Room> hotel(nb_room);
22
23      while(true)
24      {
25          cout << " Add Client(A), Empty Room(E), Display (D): ";
26          char command;
27          cin >> command;
28          switch(command)
29          {
30            case 'A': addClient(client,hotel);
31                      break;
32            case 'E': emptyRoom(client,hotel);
33                      break;
34            case 'D': displayClient(client);
35                      break;
36            default:  cout << endl << "End of program" << endl;
37                      return 0;
38          }
39      }
40      return 0;
41  }
42
43  // Display all client names in the terminal
44  void displayClient(const vector<string>& client)
45  {
46      cout << endl;
47      for(auto c : client)
48          cout << c << endl;
49  }
50
51
52  // Read a room number in [0, 9]
53  void readRoomNumber(unsigned int& roomNumber)
54  {
55      do
56      {
57          cout << "Provide a Room Number in [0, 9]: " ;
58          cin  >> roomNumber;
59      } while(roomNumber > 9);
60  }
61
62  // Suite du code sur la page suivante / Please read next page
```

```
1   // Read a client name (supposed to be a single word) and a room
2   // number. Then update the client list and hotel Room if the
3   // room is not currently occupied
4   void addClient(vector<string>& client, vector<Room>& hotel)
5   {
6       string clientName("");
7       cout  << "Provide client name (single word): " <<endl;
8       cin   >> clientName;
9
10      unsigned int roomNumber(0);
11      readRoomNumber(roomNumber);
12
13      if(hotel[roomNumber].occupied != true)
14      {
15          hotel[roomNumber].occupied = true;
16          hotel[roomNumber].client   = clientName;
17          client.push_back(clientName);
18      }
19      else
20      {
21          cout << "Sorry but this room is occupied" << endl;
22      }
23  }
24
25  // Read the room number to liberate ; if this room is occupied,
26  // remove the client name from the list of clients
27  // and update the hotel Room.
28  void emptyRoom(vector<string>& client, vector<Room>& hotel)
29  {
30      unsigned int roomNumber(0);
31      readRoomNumber(roomNumber);
32
33      if(hotel[roomNumber].occupied == true)
34      {
35          hotel[roomNumber].occupied = false;
36          hotel[roomNumber].client="";
37          client.pop_back();
38      }
39      else
40      {
41          cout << "This room is NOT occupied" << endl;
42      }
43  }
```

5.2) Maintenant on s'intéresse à un contexte pour lequel l'erreur sémantique n'a aucun impact (c'est comme si elle n'existait pas). Pour cela indiquer les actions effectuées par le programme lorsque le fichier **test1.txt** est redirigé sur l'entrée par défaut.

Inutile de ré-écrire les messages d'invitation [W]

Contenu de test1.txt	Action effectuée…	…et affichage ***résultant*** éventuel
A DUBOIS 5	**Ajouter DUBOIS**, mise à jour de **client** et **hotel** (chambre 5)	
D	**Display**/Affiche le contenu de **client**	**DUBOIS**
E 5	**Enlever** le **client** de la chambre 5, mise à jour de **hotel** et **client**	
D	**Display**/ Affiche le contenu de **client** qui est vide	*rien n'est affiché car* **client** *est vide*
Q	Quitte le programme	**End of program**

TEST2 Sem1 MT & EL 22.11.2019 EPFL / R. Boulic

W=blanc, B=bleu, S=saumon, Y=jaune, All = même réponse pour tous

Inutile de ré-écrire les messages d'invitation [B]

Contenu de test1.txt	**Action effectuée…**	**…et affichage *résultant* éventuel**
A DUROC 5	**Ajouter DUROC**, mise à jour de **client** et **hotel** (chambre 5)	
D	**Display**/Affiche le contenu de **client**	**DUROC**
E 5	**Enlever** le **client** de la chambre 5, mise à jour de **hotel** et **client**	
D	**Display**/ Affiche le contenu de **client** qui est vide	*rien n'est affiché car* **client** *est vide*
Q	Quitte le programme	**End of program**

Inutile de ré-écrire les messages d'invitation [W]

Contenu de test1.txt	**Action effectuée…**	**…et affichage *résultant* éventuel**
A DUTRONC 5	**Ajouter DUTRONC**, mise à jour de **client** et **hotel** (chambre 5)	
D	**Display**/Affiche le contenu de **client**	**DUTRONC**
E 5	**Enlever** le **client** de la chambre 5, mise à jour de **hotel** et **client**	
D	**Display**/ Affiche le contenu de **client** qui est vide	*rien n'est affiché car* **client** *est vide*
Q	Quitte le programme	**End of program**

Inutile de ré-écrire les messages d'invitation [W]

Contenu de test1.txt	**Action effectuée…**	**…et affichage *résultant* éventuel**
A DUPONT 5	**Ajouter DUPONT**, mise à jour de **client** et **hotel** (chambre 5)	
D	**Display**/Affiche le contenu de **client**	**DUPONT**
E 5	**Enlever** le **client** de la chambre 5, mise à jour de **hotel** et **client**	
D	**Display**/ Affiche le contenu de **client** qui est vide	*rien n'est affiché car* **client** *est vide*
Q	Quitte le programme	**End of program**

5.2) C'est maintenant que le contexte révèle l'erreur sémantique recherchée. Pour documenter le problème, indiquer l'action effectuée par le programme lorsque le fichier **test2.txt** est redirigé sur l'entrée par défaut.

Inutile de ré-écrire les messages d'invitation [W]

Contenu de test2.txt	**Action du programme…**	**…et affichage *résultant* éventuel**
A DUMONT 0	**Ajouter DUMONT**, mise à jour de **client** et **hotel** (chambre 0)	
A DUVAL 8	**Ajouter DUVAL**, mise à jour de **client** et **hotel** (chambre 8)	
A DURAND 3	**Ajouter DURAND**, mise à jour de **client** et **hotel** (chambre 3)	
D	**Display/**Affiche le contenu de client	**DUMONT** **DUVAL** **DURAND**
E 3	**Enlever** le **client** de la chambre 3, mise à jour de **hotel** et **client**	
D	**Display/**Affiche le contenu de client	**DUMONT** **DUVAL**
A DUPRAZ 2	**Ajouter DUPRAZ**, mise à jour de **client** et **hotel** (chambre 2)	
E 8	**Enlever** le **client** de la chambre 8, mise à jour de **hotel** et **client**	
D	**Display/**Affiche le contenu de client	**DUMONT** **DUVAL**
Q	**Quitte le programme**	**End of program**

Inutile de ré-écrire les messages d'invitation [B]

Contenu de test2.txt	**Action du programme…**	**…et affichage *résultant* éventuel**
A DUPIC 0	**Ajouter DUPIC**, mise à jour de **client** et **hotel** (chambre 0)	
A DUMONT 3	**Ajouter DUMONT**, mise à jour de **client** et **hotel** (chambre 3)	
A DUVAL 8	**Ajouter DUVAL**, mise à jour de **client** et **hotel** (chambre 8)	
D	**Display/**Affiche le contenu de client	**DUPIC** **DUMONT** **DUVAL**
E 8	**Enlever** le **client** de la chambre 8, mise à jour de **hotel** et **client**	
D	**Display/**Affiche le contenu de client	**DUPIC** **DUMONT**
A DURAND 6	**Ajouter DURAND**, mise à jour de **client** et **hotel** (chambre 6)	
E 3	**Enlever** le **client** de la chambre 3, mise à jour de **hotel** et **client**	
D	**Display/**Affiche le contenu de client	**DUPIC** **DUMONT**
Q	**Quitte le programme**	**End of program**

Inutile de ré-écrire les messages d'invitation [S]

Contenu de test2.txt	Action du programme…	…et affichage *résultant* éventuel
A DUPRAZ 0	**Ajouter DUPRAZ**, mise à jour de **client** et **hotel** (chambre 0)	
A DUPIC 6	**Ajouter DUPIC**, mise à jour de **client** et **hotel** (chambre 6)	
A DUMONT 2	**Ajouter DUMONT**, mise à jour de **client** et **hotel** (chambre 2)	
D	**Display/**Affiche le contenu de client	**DUPRAZ** **DUPIC** **DUMONT**
E 2	**Enlever** le **client** de la chambre 2, mise à jour de **hotel** et **client**	
D	**Display/**Affiche le contenu de client	**DUPRAZ** **DUPIC**
A DUVAL 8	**Ajouter DUVAL**, mise à jour de **client** et **hotel** (chambre 8)	
E 6	**Enlever** le **client** de la chambre 6, mise à jour de **hotel** et **client**	
D	**Display/**Affiche le contenu de client	**DUPRAZ** **DUPIC**
Q	**Quitte le programme**	**End of program**

Inutile de ré-écrire les messages d'invitation [Y]

Contenu de test2.txt	Action du programme…	…et affichage *résultant* éventuel
A DUVAL 0	**Ajouter DUVAL**, mise à jour de **client** et **hotel** (chambre 0)	
A DUPRAZ 4	**Ajouter DUPRAZ**, mise à jour de **client** et **hotel** (chambre 4)	
A DUPIC 7	**Ajouter DUPIC**, mise à jour de **client** et **hotel** (chambre 7)	
D	**Display/**Affiche le contenu de client	**DUVAL** **DUPRAZ** **DUPIC**
E 7	**Enlever** le **client** de la chambre 7, mise à jour de **hotel** et **client**	
D	**Display/**Affiche le contenu de client	**DUVAL** **DUPRAZ**
A DUMONT 2	**Ajouter DUMONT**, mise à jour de **client** et **hotel** (chambre 2)	
E 4	**Enlever** le **client** de la chambre 4, mise à jour de **hotel** et **client**	
D	**Display/**Affiche le contenu de client	**DUVAL** **DUPRAZ**
Q	**Quitte le programme**	**End of program**

Où se trouve l'erreur sémantique dans le code (fonction + N° ligne) : emptyRoom ligne 37

Pourquoi ? pop_back() enlève le dernier client ajouté qui n'est pas forcément celui qu'il faut enlever

Que faudrait-il faire pour la corriger :

1) d'abord déterminer quel élément **i** du vector **client** est celui qui de la chambre **roomNumber**

2) ensuite échanger la valeur des éléments **i** et **size()-1** du vector **client**

3) seulement après les 2 étapes précédentes, on peut faire l'appel à **pop_back()** sur **client**.