

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

1) (4 pts) vector : compléter un programme de calcul d'une moyenne mobile [All]

On veut écrire un programme `moy.cc` qui contient une boucle qui 1) lit un entier `int` à chaque passage, 2) range cette valeur dans le vector `v` (détails en 1.2) et affiche la moyenne des valeurs contenues dans `v`. La structure du programme est visible ci-dessous ; il faut seulement compléter la fonction `main()`.

1.1) Lecture : il faut quitter le programme dès qu'il y a un échec de la lecture de l'entier `val` à cause d'une détection de type `EOF`. Max en 2 lignes.

1.2) Ranger val : En cas de succès de la lecture de l'entier `val`, il faut le ranger dans le **vector v** en respectant la contrainte suivante : le **vector v** est vide au début et grandit quand on ajoute une valeur entière MAIS **sa taille ne doit jamais dépasser 5 éléments**. C'est pourquoi dès que la taille atteint 5, l'indice `i` de rangement des valeurs doit être géré pour que la dernière valeur lue remplace la valeur la plus ancienne du **vector**. Environ 5-7 lignes.

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4  void affiche_moy(vector<int>& v);
5
6  int main()
7  {
8      vector<int> v;
9      int i(0);
10     while(true)
11     {
12         int val(0) ;
13         // quitter le programme en cas de lecture de EOF
14         cin >> val ;
15         if(cin.eof()) return 0 ;
16
17         // ranger val au bon endroit dans v
18         if(v.size() < 5) v.push_back(val) ;
19         else
20         {
21             v[i%5] = val;
22             ++i;
23         }
24         afficher_moy(v) ;
25     }
26     return 0;
27 }
28 void affiche_moy(vector<int>& v)
29 {
30     if(v.size()>0)
31     {
32         double sum(0.);
33         for(auto val : v) sum += val;
34         cout << sum/v.size() << endl;
35     }
36 }

```

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

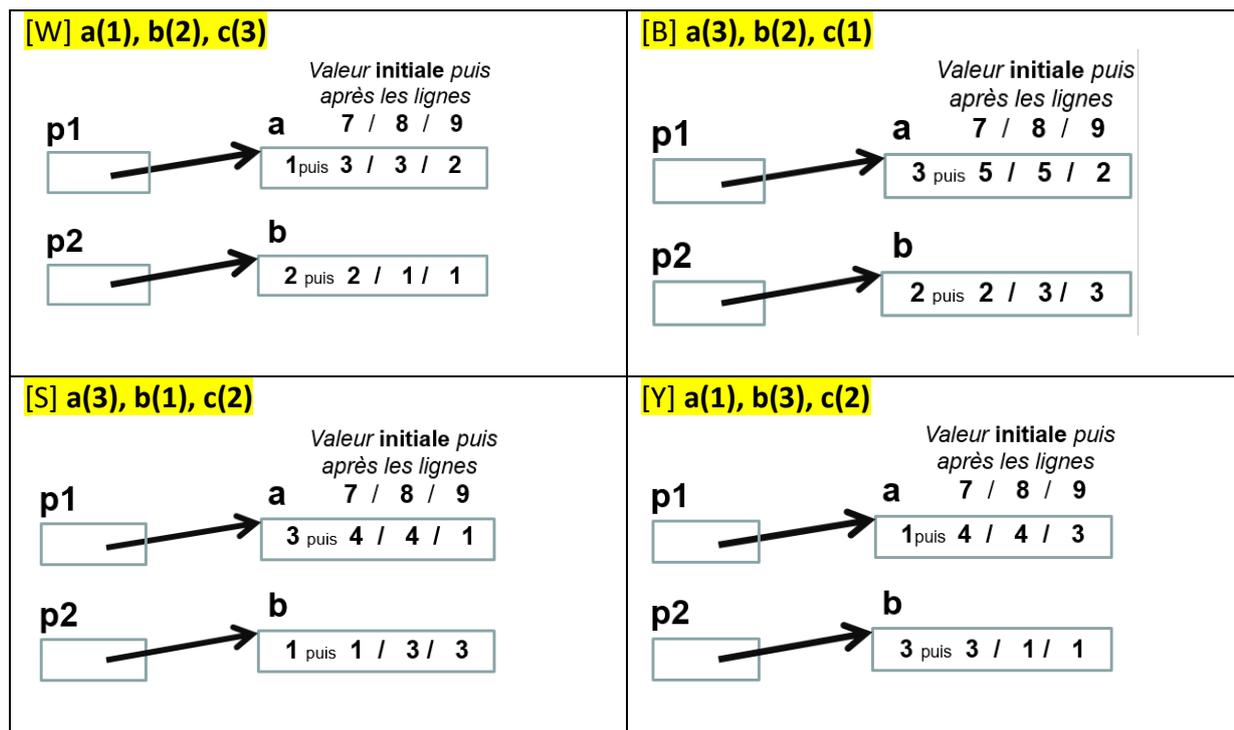
2) (5 pts) Pointeurs et variables

Ce programme compile en C++11 et s'exécute correctement

```

1  #include <iostream>
2
3  using namespace std;
4
5  void foo(int *p1, int *p2)
6  {
7      *p1 = *p1 + *p2;
8      *p2 = *p1 - *p2;
9      *p1 = *p1 - *p2;
10 }
11
12 int main()
13 {
14     int a(1), b(2), c(3);
15
16     foo(&a, &b);
17     foo(&b, &c);
18     foo(&c, &a);
19
20     cout << a << " " << b << " " << c << endl;
21
22     return 0;
23 }
    
```

2.1) Dessiner avec la représentation des boîtes et flèches les actions successives effectuées pendant l'appel de la fonction `foo(&a, &b)` à la ligne 16. Montrer les pointeurs `p1` et `p2`, les variables pointées et les valeurs qu'elles prennent pendant l'exécution des lignes 7 à 9 :



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

2.2) Compléter ensuite le tableau :

[W]

Valeur de ... Après l'exécution de la ligne...	a	b	c
14	1	2	3
16	2	1	3
17	2	3	1
18	1	3	2

[B]

Valeur de ... Après l'exécution de la ligne...	a	b	c
14	3	2	1
16	2	3	1
17	2	1	3
18	3	1	2

[S]

Valeur de ... Après l'exécution de la ligne...	a	b	c
14	3	1	2
16	1	3	2
17	1	2	3
18	3	2	1

[Y]

Valeur de ... Après l'exécution de la ligne...	a	b	c
14	1	3	2
16	3	1	2
17	3	2	1
18	1	2	3

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

3) (5 pts) Pointeurs et vector

Le code suivant compile en C++11 et s'exécute correctement.

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  void affiche_vecteur(vector<char> &v)
6  {
7      for(size_t i=0; i<v.size(); i++) cout << v[i];
8      cout << endl;
9  }
10
11 int main()
12 {
13     vector<char> v = {'B', 'o', 'n', 'j', 'o', 'u', 'r', '\0'};
14     char *p = &(v[0]);
15
16     cout << sizeof(v[0]) << endl;
17
18     cout << ( (*p == v[0]) ? "V" : "F" ) << endl;
19
20     cout << v[3] << endl;
21
22     cout << *p << endl;
23
24     cout << *(p+1) << endl;
25
26     cout << static_cast<char> (*p-1) << endl;
27
28     cout << static_cast<char> (*(p+3)+5) << endl;
29
30     cout << (&p[4] - &p[1]) << endl;
31
32     *(p+2) = '_';
33     cout << "v = ";
34     affiche_vecteur(v);
35
36     (*p)++;
37     cout << "v = ";
38     affiche_vecteur(v);
39
40     (*(p+4))++;
41     cout << "v = ";
42     affiche_vecteur(v);
43
44     return 0;
45 }
46
```

Remplir le tableau sur la page suivante en justifiant brièvement chaque réponse

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

[W] `vector<char> v = {'B', 'o', 'n', 'j', 'o', 'u', 'r', '\0'};`

Ligne de code	Résultat de l'affichage	Explication, soyez bref
16	1	taille d'un élément du vector v = un char = 1 octet
18	V	La char pointé par p est le premier char du vector, donc la condition est vraie
20	j	char d'indice 3 du vector
22	B	Premier char du vector pointé par p
24	o	Second char du vector pointé par p
26	A	char qui précède dans l'alphabet la première lettre du vector
28	o	Char 5 lettres plus loin dans l'alphabet que le char d'indice 3 du vector (donné à la ligne 20)
30	3	Différence des adresses exprimée en nombre de char
34	Bo_jour	Vector v dont le char d'indice 2 a été remplacé par le char '_'
38	Co_jour	Vector v avec incrémentation du code ASCII du premier char
42	Co_jpur	Vector v avec incrémentation du char d'indice 4

[B] `vector<char> v = {'B', 'a', 'n', 'j', 'o', 'e', 's', '\0'};`

Ligne de code	Résultat de l'affichage	Explication, soyez bref
16	1	taille d'un élément du vector v = un char = 1 octet
18	V	La char pointé par p est le premier char du vector, donc la condition est vraie
20	j	char d'indice 3 du vector
22	B	Premier char du vector pointé par p
24	a	Second char du vector pointé par p
26	A	char qui précède dans l'alphabet la première lettre du vector
28	o	Char 5 lettres plus loin dans l'alphabet que le char d'indice 3 du vector (donné à la ligne 20)
30	3	Différence des adresses exprimée en nombre de char
34	Ba_joes	Vector v dont le char d'indice 2 a été remplacé par le char '_'
38	Ca_joes	Vector v avec incrémentation du code ASCII du premier char
42	Ca_jpes	Vector v avec incrémentation du char d'indice 4

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

[S] `vector<char> v = {'B','i','c','o','l','o','r','\0'};`

Ligne de code	Résultat de l'affichage	Explication, soyez bref
16	1	taille d'un élément du vector v = un char = 1 octet
18	V	La char pointé par p est le premier char du vector, donc la condition est vraie
20	o	char d'indice 3 du vector
22	B	Premier char du vector pointé par p
24	i	Second char du vector pointé par p
26	A	char qui précède dans l'alphabet la première lettre du vector
28	t	Char 5 lettres plus loin dans l'alphabet que le char d'indice 3 du vector (donné à la ligne 20)
30	3	Différence des adresses exprimée en nombre de char
34	Bi_olor	Vector v dont le char d'indice 2 a été remplacé par le char '_'
38	Ci_olor	Vector v avec incrémentation du code ASCII du premier char
42	Ci_omor	Vector v avec incrémentation du char d'indice 4

[W] `vector<char> v = {'B','e','d','r','o','o','m','\0'};`

Ligne de code	Résultat de l'affichage	Explication, soyez bref
16	1	taille d'un élément du vector v = un char = 1 octet
18	V	La char pointé par p est le premier char du vector, donc la condition est vraie
20	r	char d'indice 3 du vector
22	B	Premier char du vector pointé par p
24	e	Second char du vector pointé par p
26	A	char qui précède dans l'alphabet la première lettre du vector
28	w	Char 5 lettres plus loin dans l'alphabet que le char d'indice 3 du vector (donné à la ligne 20)
30	3	Différence des adresses exprimée en nombre de char
34	Be_room	Vector v dont le char d'indice 2 a été remplacé par le char '_'
38	Ce_room	Vector v avec incrémentation du code ASCII du premier char
42	Ce_rpom	Vector v avec incrémentation du char d'indice 4

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

4) (5 pts) Le sage a dit « Tu n'ignoreras point les warnings »

4.1) Un étudiant aime tellement peu les warnings qu'il a enlevé l'option `-Wall` de la commande de compilation. De plus, pour le code de cet exercice, il a même ajouté une option qui empêche le compilateur d'afficher un warning (nous ne vous donnerons pas cette option car c'est une mauvaise pratique).

Cependant ce warning indique en fait un sérieux problème de conception même si le code compile et produit un exécutable en C++11.

Nous allons d'abord examiner l'exécution du code sur lequel le problème de conception n'a pas d'impact (questions 4.1 et 4.2) et ensuite seulement déterminer la nature du problème pour proposer une solution (4.3).

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

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 struct Couleur
6 {
7     unsigned int rouge;
8     unsigned int vert;
9     unsigned int bleu;
10 };
11
12 struct Lampe
13 {
14     bool eclaire;
15     Couleur couleur;
16     string name;
17 };
18
19 Lampe* cree_lampe(bool, string, unsigned int,
20                 unsigned int, unsigned int);
21 void affiche_couleur_lampe(Lampe&);
22 void affiche_etat_lampe(Lampe *);
23
24 int main()
25 {
26     Lampe lampe1 = {true, Couleur{255,0,125}, "salon"};
27     cout << "Etat lampe1 dans main : ";
28     affiche_etat_lampe(&lampe1);
29
30     Lampe *lampe2 = cree_lampe(true, "couloir", 255,255,999);
31     cout << "Etat lampe2 dans main : ";
32     affiche_etat_lampe(lampe2);
33
34     Lampe *lampe3 = cree_lampe(true, "escalier", 255,0,125);
35     cout << "Etat lampe3 dans main : ";
36     affiche_etat_lampe(lampe3);
37
38     return EXIT_SUCCESS;
39 }
40
41 void affiche_etat_lampe(Lampe *lampe)
42 {
43     if( lampe == nullptr )
44     {
45         cout << "La lampe n'est pas definie !" << endl;
46         return;
47     }
48     cout << "La lampe " << lampe->name << " est ";
49     if( !lampe->eclaire )
50     {
51         cout << " eteinte" << endl;
52         return;
53     }
54     cout << "allumee" << endl ;
55     cout << "Sa couleur est :";
56     affiche_couleur_lampe(*lampe);
57 }
58
59 void affiche_couleur_lampe(Lampe &lampe)
60 {
61     cout << " R : " << lampe.couleur.rouge
62         << ",V : " << lampe.couleur.vert
63         << ",B : " << lampe.couleur.bleu
64         << endl;
65 }
66 // le code source continue sur la page suivante
```

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

```

67 // suite de la page précédente
68
69 Lampe* cree_lampe(bool eclaire, string name,
70                 unsigned int rouge,
71                 unsigned int vert,
72                 unsigned int bleu)
73 {
74     if(rouge > 255) return nullptr;
75     if(vert > 255) return nullptr;
76     if(bleu > 255) return nullptr;
77
78     if(name.size() == 0) return nullptr;
79
80     Lampe lampe;
81     lampe.name = name;
82     lampe.eclaire = eclaire;
83
84     Couleur couleur;
85     couleur.rouge = rouge;
86     couleur.vert = vert;
87     couleur.bleu = bleu;
88
89     lampe.couleur = couleur;
90
91     cout << "Initialisation de la lampe : ";
92     affiche_etat_lampe(&lampe);
93
94     return &lampe;
95 }
96

```

4.1) Exécution des lignes 26 à 28 de la fonction main(). Quel affichage est effectué par cette portion de code qui est correcte ?

[W]

Etat lampe1 dans main : La lampe salon est allumee
sa couleur est : R : 255, V : 0, B : 125

[B]

Etat lampe1 dans main : La lampe chambre est allumee
sa couleur est : R : 255, V : 0, B : 125

[S]

Etat lampe1 dans main : La lampe studio est allumee
sa couleur est : R : 255, V : 0, B : 125

[W]

Etat lampe1 dans main : La lampe cuisine est allumee
sa couleur est : R : 255, V : 0, B : 125

4.2) Exécution des lignes 30 à 32 de la fonction main(). Cette portion de code s'exécute comme prévu ; quel affichage est effectué ?

[All]

Etat lampe2 dans main : La lampe n'est pas définie !

En effet, la fonction cree_lampe détecte que la composante couleur de 999 fournie pour initialiser le champ bleu du champ couleur de la lampe n'est pas correcte ; elle renvoie donc nullptr.

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

4.3) Exécution des lignes 34 à 36 de la fonction main(). Cette portion de code ne s'exécute pas comme prévu par la personne qui a écrit le programme.

4.3.1) indiquer d'abord quel est l'affichage attendu :

[All] La personne qui a conçu le programme a probablement imaginé qu'il était possible de créer une struct **lampe** avec **creer_lampe** puis de l'utiliser ailleurs en renvoyant son adresse. L'affichage espéré était :

```
Initialisation de la lampe : la lampe escalier est allumee
Sa couleur est : R : 255, V : 0, B : 125
Etat lampe3 dans main : La lampe escalier est allumee
sa couleur est : R : 255, V : 0, B : 125
```

4.3.2) expliquer l'erreur de conception du programme :

[All] la fonction **creer_lampe** renvoie l'adresse de la variable locale **lampe** alors que celle-ci cesse d'exister à la fin de l'appel de la fonction **creer_lampe**. Une telle adresse ne doit pas être utilisée en dehors de la fonction **creer_lampe**. Heureusement le compilateur détecte ce problème et affiche un warning assez clair (pour une fois).

En fait à l'exécution ce qui est renvoyé est une adresse **nullptr** ce qui permet au programme de détecter rapidement qu'il y a un problème avec la fonction **creer_lampe**.

4.3.3) Sans écrire de code, indiquer comment il aurait fallu écrire le programme pour obtenir le comportement attendu.

[All] Si on veut écrire une fonction qui crée une variable lampe **dont l'existence n'est pas limitée par la durée de l'appel de creer_lampe** ALORS il faut faire une **allocation dynamique** d'une structure **lampe** et l'adresse obtenue peut être renvoyée par la fonction pour un usage dans main().