

Matricule :

Nom :

Prénom :

Groupe :

Code 1

```
1 #include <iostream>
2 template<class T> struct Chips {
3     T t; double f;
4     Chips(T t) { this->t = t; }
5     Chips(double f) { this->f = f; }
6     void print() { std::cout << t << " " << f << std::endl; }
7 };
8 int main() {
9     Chips<double> b(3);    b.t = 5;
10    b.print();
11 }
```



X

Affichage souhaité du Code 1

```
1 5 3
```

Affichage produit par le code 1

```
1 Erreur de compilation
```

Code 2

```
1 #include <iostream>
2 #include <vector>
3
4 template<int i , template<class> class Container , class T>
5 T get(Container<T>& c)
6 {
7     return c[i];
8 }
9
10 int main()
11 {
12     std::vector<int> v = {1,2,3,4,5};
13     get<2>(v) = 5; //vous ne pouvez pas changer cette ligne
14     std::cout << v[2];
15 }
```



X

Affichage souhaité du Code 2

```
1 5
```

Affichage produit par le code 2

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 3

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 auto cmd = "Travaille";
6 int motiv = 404;
7
8 struct Student { void work() { throw motiv; } };
9
10 struct Teacher
11 {
12     Teacher()
13     {
14         cout << cmd << endl; throw cmd;
15     }
16
17     void command(Student * s)
18     {
19         try { s->work(); }
20         catch(...)
21         {
22             cout << "Nous afons les moyens te fous faire posser !" << endl;
23         }
24     }
25 };
26
27 int main()
28 {
29     Student s;
30     Teacher* abs;
31     try { *abs = Teacher(); }
32     catch(string& ) { cout << "Oui, bien sur" << endl; }
33     catch(...)
34     {
35         cout << "Non" << endl;
36         abs->command(&s );
37     }
38 }
```



Changer auto en String pour pouvoir
rentrer dans le catch qu'on veut.

Affichage souhaité du Code 3

```
1 Travaille
2 Oui, bien sur
```

Affichage produit par le code 3

```
1 Travaille
2 Non
3 Nous afons les moyens te fous faire posser !
```

Matricule :

Nom :

Prénom :

Groupe :

Code 4

```
1 #include <iostream>
2
3 struct Number
4 {
5     double f;
6     Number(double f) : f(f) {}
7     operator int() { return f; }
8     Number operator +(Number n) { return Number(f + n.f); }
9 };
10
11 int main()
12 {
13     Number n = 3; std::cout << n + 2.5 << std::endl;
14 }
```

Dans cette ligne, on voit qu'on a $f + n.f$ donc un double + un objet number qui prend un double. Donc quand on utilise l'opérateur + avec un number, on doit respecter l'ordre $f + n.f$, ce n'est pas commutatif.

Solution : on inverse $n + 2.5$ qui devient $2.5 + n$ pour avoir l'ordre requis car la surcharge d'opérateur n'est pas commutatif.

Affichage souhaité du Code 4

1 5.5

Affichage produit par le code 4

1 Erreur de compilation

Code 5

```
1 #include <iostream>
2
3 struct Tartiflette
4 {
5     void renifler() const { std::cout << "Reniflage_constant" << std::endl; }
6     void renifler() { std::cout << "Reniflage_de_trop_près" << std::endl; }
7 };
8
9 int main()
10 {
11     Tartiflette t; t.renifler();
```

le compilateur choisit celui qui est le mieux adapté, vu que nous n'avons pas de const à tartiflette, il a choisi l'autre fonction.
- Solution : Ajouter const à tartiflette t

Affichage souhaité du Code 5

1 Reniflage constant

Affichage produit par le code 5

1 Reniflage de trop près

Matricule :

Nom :

Prénom :

Groupe :

Code 6

```
1 #include <iostream>
2 #include <vector>
3 struct Tank
4 {
5     static int count;
6     int id;
7     static std::string msg[3];
8     Tank() : id(count) { count++; }
9     ~Tank() { cout << msg[id % 3]; count--; }
10 };
11 int Tank::count = 0;
12 std::string Tank::msg[3] = {"They\u201c\u2019are\u201c\u2019Panzer\u201c\u2019elite\n",
13                         "born\u201c\u2019to\u201c\u2019compete\n", "never\u201c\u2019retreat\n"};
14
15 int main() { std::vector<Tank*> v = {new Tank(), new Tank(), new Tank()}; }
```

Comme nous avons créer les objets dynamiquement avec un new, nous avons besoin d'appeler le destructeur explicitement.
Solution : delete v[]...

Affichage souhaité du Code 6

```
1 They are the Panzer elite
2 born to compete
3 never retreat
```

Affichage produit par le code 6

faut toujours détruire explicitement avec delete les objets que l'instances avec new

Code 7

```
1 #include <iostream>
2 #include <vector>
3 struct Nyan { int i; Nyan(int i) : i(i) {} };
4
5 int main()
6 {
7     std::vector<Nyan> v(3);
8     for(Nyan n : v)
9         std::cout << n.i << "\u201c";
10 }
```



Affichage souhaité du Code 7

```
1 2 3
```

Affichage produit par le code 7

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 8

```
1 #include <iostream>
2 class A
3 {
4     protected:
5         int i;
6     public:
7         A(int i = 0) : i(i) {}
8         void print() { std::cout << i << std::endl; }
9     };
10 class B : public A
11 {
12     int j;
13     public:
14         B(int i = 0) : A(i + 2), j(i + 3) {}
15         void print() { std::cout << (i + j) << std::endl; }
16     };
17 int main() { A a = B(); a.print(); }
```

L'objet appelle les méthodes de sa classe, comme nous avons un objet A, il utilise les méthodes de la classe A.
Solution : créer un objet B pour appeler la méthode de la classe B.

Affichage souhaité du Code 8

1 5

Affichage produit par le code 8

1 2

Code 9

```
1 #include <iostream>
2 struct A { virtual A& operator=(const A&) { std::cout << "A" << "\n"; } };
3 struct B : A { virtual B& operator=(const B&) { std::cout << "B" << "\n"; } };
4 int main()
5 {
6     A * a1 = new A(); A * a2 = new A(); B * b1 = new B(); B * b2 = new B();
7     *a1 = *a2; *b1 = *b2; *a1 = *b1;
8 }
```



Affichage souhaité du Code 9

1 A B B

Affichage produit par le code 9

1 A B A

Matricule :

Nom :

Prénom :

Groupe :

Code 10

FICHIER DEPARTEMENT.CPP

```
1 #include <iostream>
2
3 class Manager;
4 class Departement
5 {
6     public:
7         Manager * b;
8         Departement() : b(nullptr) {}
9         void print() { std::cout << "Departement" << std::endl; }
10    };
11
```

FICHIER MANAGER.CPP

```
1 #include <iostream>
2
3 #include "Departement.cpp"
4
5 class Manager
6 {
7     public:
8         Departement a;
9         Manager() {}
10        void print() { std::cout << "Manager" << std::endl; }
11    };
12
```

FICHIER MAIN.CPP

```
1 #include <iostream>
2 #include "Departement.cpp"
3 #include "Manager.cpp"
4
5 int main()
6 {
7     Departement b; Manager a;
8     b.b = &a; a.a = b;
9     a.print(); b.print();
10 }
```

Régler le problème des
inclusions multiples en
implémentant
#ifndef/#define et utilise
une référence pour
l'attribut département
dans
la classe manager

Affichage souhaité du Code 10

```
1 Manager
2 Departement
```

Affichage produit par le code 10

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 11

```
1 #include <iostream>
2
3 struct A
4 {
5     int * tab; int n;
6     A(int i) : tab(new int[i]), n(i) {}
7     ~A() { delete tab; }
8     void print()
9     {
10        for(int i = 0; i < n; i++)
11            std::cout << tab[i] << " ";
12    }
13};
14
15 void f(A b) { std::cout << "Symphony_of_destruction" << std::endl; }
16
17 int main()
18 {
19     A a(3); a.tab[0] = 0; a.tab[1] = 1; a.tab[2] = 2;
20     f(a);
21     a.print();
22
23     A aa(2); a.tab[0] = 3; a.tab[1] = 4;
24     aa = a;
25     aa.print();
26
27     aa = aa;
28     aa.print();
29 }
```



Affichage souhaité du Code 11

```
1 Symphony of destruction
2 0 1 2
3 0 1 2
4 0 1 2
```

Affichage produit par le code 11

```
1 Erreur de segmentation
```

Code 1

```
1 #include <iostream>
2 using namespace std;
3
4 class A
{
5     public:
6         void f(int n) {cout << "A::integer " << n << endl; }
7         void f(char n) {cout << "A::character " << n << endl; }
8 };
9
10
11 class B : public A
12 {
13     public:
14         void f(int n, int m) { cout << "B:: integers " << n << " " << m << endl; }
15 };
16
17
18 int main()
19 {
20     int n = 1;
21     char c = 'a';
22     B b;
23     b.f(n);
24     b.f(c);
25     b.f(n, c);
26 }
```

Problème de surdéfinition, le compilateur choisit tjs les fonctions de

la classe dérivés, comme celle de la classe dérivés prend 2 paramètres, dans le main il en manque un, on a donc une erreur.

Solution : Surdéfinir les 2 fonctions de classe de base A dans la classe dérivés B.

Solution 2 : Faire du transtypage en faisant appelle à la classe mère de l'objet donc b.Mere::f(n)

Affichage souhaité du Code 1

```
1 A::integer
2 A::character
3 B:: integers
```

Affichage produit par le code 1

```
1 Erreur de compilation
```

Code 2

```
1 #include <iostream>
2
3 int f() {
4     static int i { 4 };
5     return --i;
6 }
7
8 int main()
9 {
10    std::cout << f() << std::endl;
11    while (f()) { }
12    std::cout << f() << std::endl;
13 }
```

Remplacer la pre
incrementation par un post
incrementation donc i--

Affichage souhaité du Code 2

```
1 4
2 -1
```

Affichage produit par le code 2

```
1 3
2 -1
```

Code 3

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Integer
6 {
7     unsigned i;
8     bool positive;
9
10    public:
11        Integer(unsigned i, bool positive = true) : i(i), positive(positive) {}
12        Integer operator +(Integer a)
13        {
14            if(positive && a.positive)
15                return Integer(i + a.i);
16            else if(positive && !a.positive)
17                return Integer(i - a.i);
18            else if(!positive && a.positive)
19                return Integer(a.i - i);
20            else
21                return Integer(-i - a.i);
22        }
23
24        void print()
25        {
26            if(!positive)
27                cout << "-";
28            cout << i << endl;
29        }
30    };
31
32    int main()
33    {
34        Integer a1(2u);
35        Integer a2(3u, false);
36
37        Integer s = a1 + a2;
38        s.print();
39    }
```

Remplacer l'entier non signé
par un entier pour qu'on
puisse obtenir une valeur de
retour négative

Affichage souhaité du Code 3

```
1 -1
```

Affichage produit par le code 1

```
1 4294967295
```

Code 4

```
1 #include <iostream>
2 using namespace std;
3
4 class exceptA {};
5 class exceptB : public exceptA {};
6 class exceptC : public exceptB {};
7
8 void f() { throw exceptB(); }
9
10 int main()
11 {
12     try
13     {
14         f();
15     }
16     catch(exceptA& e)
17     {
18         cout << "I caught an A" << endl;
19     }
20     catch(exceptB& e)
21     {
22         cout << "I caught a B" << endl;
23     }
24     catch(exceptC& e)
25     {
26         cout << "I caught a C" << endl;
27     }
28     catch(...)
29     {
30         cout << "I caught something" << endl;
31     }
32 }
```

Changer l'ordre des catches en commençant par C puis B puis A puisque qu'il y a héritage et qu'il considère que l'exception exceptB comme un except A puisqu'il hérite de A

Affichage souhaité du Code 4

```
1 I caught a B
```

Affichage produit par le code 4

```
1 I caught an A
```

Code 5

```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 class A
7 {
8     public:
9         int i;
10        A(int i) : i(i) {}
11        void print() { cout << i << " "; }
12    };
13
14 int main()
15 {
16     vector<A> v(5);
17     for(int j = 0; j < v.size(); j++)
18         v[j].i = j;
19     for(A a : v)
20         a.print();
21 }
```

Ajouter un constructeur par défaut puisque qu'on se retrouve avec un vector de A qui possède des valeurs non instanciées

Affichage souhaité du Code 5

```
1 0 1 2 3 4
```

Affichage produit par le code 5

```
1 Erreur de compilation
```

Code 6

```
1 #include <iostream>
2 using namespace std;
3 class A {
4     public:
5         A() { cout << "+A" << endl; }
6         A(const A& a) { cout << "rA" << endl; }
7         virtual ~A() { cout << "-A" << endl; }
8     };
9 class B {
10    public:
11        B() { cout << "+B" << endl; }
12        B(const B& b) { cout << "rB" << endl; }
13        virtual ~B() { cout << "-B" << endl; }
14    };
15 class C : public A, public virtual B {
16    public:
17        C() { cout << "+C" << endl; }
18        C(const C& c) { cout << "rC" << endl; }
19        virtual ~C() { cout << "-C" << endl; }
20    };
21
22 void f(A a) {}
23
24 int main() {
25     C c; cout << endl;
26     f(c); cout << endl;
27     C * cc = new C(); cout << endl;
28     delete cc; cout << endl;
29 }
```

Problème venant de l'ordre de dérivation des constructeurs et destructeurs qui change, le destructeur de la classe dérivé C a priorité sur la classe de base A et B

Solution : retirer les virtual

Affichage souhaité du Code 6

```
1 +A   rA   +A   -C   -C
2 +B   -A   +B   -B   -B
3 +C       +C   -A   -A
```

Affichage produit par le code 6

```
1 +B   rA   +B   -C   -C
2 +A   -A   +A   -A   -A
3 +C       +C   -B   -B
```

Remarque 1. Les affichages ci-dessus sont à lire en colonnes, de haut en bas et de gauche à droite.

Code 7

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     unsigned i = 5;
8     while(i >= 0)
9     {
10         if(i != 0)
11             cout << i << endl;
12         i--;
13     }
14     cout << "BOOM" << endl;
15 }
```

Comme c'est un unsigned, il ne quitte jamais la boucle puisque la valeur de i n'est jamais en dessous de zero

Solution : remplacer par un int

Affichage souhaité du Code 7

```
1 5
2 4
3 3
4 2
5 1
6 BOOM
```

Affichage produit par le code 7

```
1 Boucle infinie
```

Code 8

```
1 #include <iostream>
2
3 void swap(int * i, int * j)
4 {
5     int * tmp = i ;
6     i = j ; j = tmp ;
7 }
8
9 int main()
10 {
11     int i = 2; int j = 3; swap(&i , &j );
12     cout << i << " " << j << endl ;
13 }
```

Remplacer les pointeurs par des references en parametres afin de changer leurs valeurs directement dans la memoire

Affichage souhaité du Code 8

```
1 3 2
```

Affichage produit par le code 8

```
1 2 3
```

Code 9

```
1 #include <iostream>
2 class A {};
3 class B {
4     A * a;
5     public:
6     B() : a(new A()) {}
7     ~B() { delete a; }
8     void print() { cout << "Get_to_the_choppaaa" << endl; }
9 };
10
11 void f(B b) { std::cout << "You_have_no_respect_for_logic" << std::endl; }
12
13 int main()
14 {
15     B b;
16     f(b);          Mettre une reference afin
17     b.print();      d'éviter de faire une copie de
18                 la valeur
19     B bb;
20     bb = b;
21     bb.print();
22
23     bb = bb;        Ecrire une constructeur de
24     bb.print();    copie du coup un opérateur
                     d'affectation qui vérifie si la
                     valeur affectée n'est pas
                     identique à celle reçue en
                     paramètre
25 }
```

Affichage souhaité du Code 9

```
1 You have no respect for logic
2 Get to the choppaaa
3 Get to the choppaaa
4 Get to the choppaaa
```

Affichage produit par le code 9

```
1 Erreur de segmentation
```

Code 10

```
1 #include <iostream>
2 #include <list>
3
4 int main()
5 {
6     std::list<int> l;
7     for (int i = 0; i <= 6; i++)
8         l.push_back(i * i - i);
9     for (int i = 0; i < l.size(); i++)
10    {
11        auto j = l.rend();
12        std::cout << *j << " ";
13        l.pop_back();
14    }
15 }
```

Comme on fait un pop back la taille du vector fait que se réduire et interrompt donc le parcours du for de maniere prematuré, il faut donc stocker la taille du vector dans un int afin de preserver la taille et il faut savoir que le rend est un iterateur inverse

Affichage souhaité du Code 10

```
1 30 20 12 6 2 0 0
```

Affichage produit par le code 10

```
1 30 20 12 6
```

Code 11

```
1 #include <iostream>
2
3 class A
4 {
5     int i;
6     public:
7     A operator +(int i) { std::cout << "For Macrage we march" << std::endl; }
8     A operator +(A a) { std::cout << "and we shall know no fear" << std::endl; }
9 };
10
11 int main()
12 {
13     int i1; A a1; A a2;
14     i1 + a1 + a2; //vous ne pouvez pas changer cette ligne
15 }
```

Ajouter une surcharge d'operator de int afin de prendre en compte les operations avec des objets A pour les entiers

Affichage souhaité du Code 11

```
1 For Macrage we march
2 and we shall know no fear
```

Affichage produit par le code 11

```
1 Erreur de compilation
```

Code 13

```
1 #include <iostream>
2 #include <list>
3 #include <algorithm>
4
5 class MyList
6 {
7     std::list<int> l;
8     public:
9         MyList(int i = 0) : l(std::list<int>(i)) {}
10        void add(int i) { l.push_back(i); }
11        void print()
12        {
13            for(int i : l)
14                std::cout << i << " ";
15            std::cout << std::endl;
16        }
17    };
18
19 int main()
20 {
21     std::list<MyList<int>> list(5);
22     for(auto it = list.begin(), int n = 4; it != list.end(); it++, n--)
23     {
24         MyList<int> l;
25         for(int i = n; i >= 0; i--)
26             l.add(i);
27         *it = l;
28     }
29     std::sort(list.begin(), list.end()); // trie les listes par taille croissante
30     for(MyList<int> l : list)
31         l.print();
32 }
```

On ne peut pas comparer
deux listes du coup il faut
faire la surcharge d'opérateur
de comparaison

Affichage souhaité du Code 13

```
1 0
2 0 1
3 0 1 2
4 0 1 2 3
5 0 1 2 3 4
```

Affichage produit par le code 13

```
1 Erreur de compilation
```

Code 14

```
1 #include <iostream>
2
3 class A
4 {
5     protected:
6         int i;
7     public:
8         A(int i = 0) : i(i) {}
9         void print() { std::cout << i << std::endl; }
10    };
11
12 class B : public A
13 {
14     int j;
15     public:
16         B(int i = 0) : A(i + 2), j(i + 3) {}
17         void print() { std::cout << (i + j) << std::endl; }
18    };
19
20 int main() { A a = B(); a.print(); }
```

Comme A a convertit le B en A, il considère mon nouvel objet B comme un objet A qui ne prend en compte que l'attribut i

La solution est de créer un pointeur de A qui reçoit l'adresse d'un objet B alloué dynamiquement par un new B();

Affichage souhaité du Code 14

1 5

Affichage produit par le code 14

1 2

Code 17

```
1 #include <iostream>
2 using namespace std;
3 class A {
4     public:
5         A() { cout << "+A" ; }
6         A(const A& a) { cout << "rA" ; }
7         ~A() { cout << "-A" ; }
8         A& operator =(const A& a) { cout << "=A" << endl; return *this; }
9     };
10
11 void f(A a) {}
12 void g(A& a) {}
13
14 int main()
15 {
16     A a;
17     f(a);
18     A * aa = new A();
19     aa = &a;
20     g(*aa);
21 }
```

Lorsqu'on fait une allocation dynamique de manier explicite, il faut détruire explicitement l'objet avec un delete, l'appelle du destructeur est manquant

Affichage souhaité du Code 17

```
1 +A rA -A +A -A -A
```

Affichage produit par le code 17

```
1 +A rA -A +A -A
```

Code 18

```
1 #include <iostream>
2 using namespace std ;
3
4 int main()
{
5     int a1 = 2.1;
6     cout << a1 << " " ;
7     int a2(2.1);
8     cout << a2 << " " ;
9     int a3{2.1};
10    cout << a3 << endl ;
11}
12
```

Il n'y a pas de conversion implicite avec la variable a3 du coup soit on met un entier soit on change son type en double ou on change son affectation par l'opérateur =

Affichage souhaité du Code 18

```
1 2 2 2
```

Affichage produit par le code 18

```
1 Erreur de compilation
```

Question 2. Décrivez les différentes classes d’allocations de variables en C++, c'est-à-dire, entre autres,

- en quoi elles se différencient d'un point de vue gestion de la mémoire,
- comment les mettre en œuvre (mots clés associés),
- la durée de vie, portée, création et destruction de la mémoire allouée,
- les mécanismes que certaines de ces classes rendent possibles, etc.

/10

Question 3. Expliquez la manière dont les chaînes de caractères sont mises en œuvre en C pur (sans les mécanismes inhérents au C++), ainsi que la façon de les manipuler.

/10

Question 4. Vous devez stocker dans un conteneur 100 000 objets de type A. Pour l'application que vous développez, vous serez amenés à régulièrement extraire (obtenir sans supprimer) des éléments à des positions arbitraires de ce conteneur. Sur quel type de conteneur votre choix se porterait-il ? *Justifiez rigoureusement votre réponse.*

/10

Code 1

```

1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     unsigned i = 5; //vous ne pouvez pas changer cette ligne
8     while(i >= 0)
9     {
10         if(i != 0) 
11             cout << i << endl;
12         i--;
13     }
14     cout << "BOOM" << endl;
15 }
```

Affichage souhaité du code 1

```

1 5
2 4
3 3
4 2
5 1
6 BOOM
```

Affichage produit par le code 1

```
1 Boucle infinie
```

Code 2

```

1 #include <iostream>
2
3 void swap(int * i, int * j)
4 {
5     int * tmp = i;
6     i = j; j = tmp; 
7 }
8
9 int main()
10 {
11     int i = 2; int j = 3; swap(&i, &j);
12     cout << i << " " << j << endl;
13 }
```

Affichage souhaité du code 2

```
1 3 2
```

Affichage produit par le code 2

```
1 2 3
```

Nom :

Prénom :

Groupe :

Matricule :

Code 3

```
1 #include <iostream>
2 class A {};
3 class B {
4     A * a;
5     public:
6     B() : a(new A()) {}
7     ~B() { delete a; }
8     void print() { cout << "Talk_to_the_hand" << endl; }
9 }
10
11 void f(B b) { std::cout << "I_lied" << std::endl; }
12
13 int main()
14 {
15     B b;
16     f(b);
17     b.print(); 
18
19     B bb;
20     bb = b;
21     bb.print();
22
23     bb = bb;
24     bb.print();
25 }
```

Affichage souhaité du code 3

```
1 I_lied
2 Talk_to_the_hand
3 Talk_to_the_hand
4 Talk_to_the_hand
```

Affichage produit par le code 3

```
1 Erreur de segmentation
```

Code 4

```
1 #include <iostream>
2 int main() {
3     int i = 5; int j = 2; double d = 1.25; //vous ne pouvez pas modifier cette ligne
4     std::cout << (j / i + d) << std::endl;
5 }
```

Solution : static cast i et j 

Affichage souhaité du code 4

```
1 3.75
```

Affichage produit par le code 4

```
1 3.25
```

Nom :

Prénom :

Groupe :

Matricule :

Code 5

```
1 #include <iostream>
2 #include <list>
3
4 int main()
5 {
6     std::list<int> l;
7     for(int i = 0; i <= 6; i++)
8         l.push_back(i * i - i);
9     for(int i = 0; i < l.size(); i++)
10    {
11        auto j = l.rend(); iterateur inverser,
12        std::cout << *j << " " ; commence par la fin
13        l.pop_back();
14    }
15 }
```

Solution : créer un entier qui stocke la taille de la liste de manière statique

quand tu pop_back tu supprimes le dernier élément mais tu réduis également la taille de la liste

Affichage souhaité du code 5

```
1 30 20 12 6 2 0 0
```

Affichage produit par le code 5

```
1 30 20 12 6
```

Code 6

```
1 #include <iostream>
2
3 class A
4 {
5     int i;
6     public:
7     A operator +(int i) { std::cout << "I_am_one_with_the_force" << std::endl; }
8     A operator +(A a) { std::cout << "and_the_force_is_with_me" << std::endl; }
9 };
10
11 int main()
12 {
13     int i1; A a1; A a2;
14     i1 + a1 + a2;
```



Affichage souhaité du code 6

```
1 I am one with the force
2 and the force is with me
```

Affichage produit par le code 6

```
1 Erreur de compilation
```

Nom :

Prénom :

Groupe :

Matricule :

Code 7

```
1 #include <iostream>
2
3 template<class T> class Brol
4 {
5     public:
6         T & t; int & i;
7         Brol(T & t) { this->t = t; }
8         Brol(int & i) { this->i = i; }
9         void print() { std::cout << t << " " << i << std::endl; }
10    };
11
12 int main()
13 {
14     Brol<int> b(2);    b.t = 3;
15     b.print();
16 }
```

X

Affichage souhaité du code 7

```
1 3 2
```

Affichage produit par le code 7

```
1 Erreur de compilation
```

Code 8

```
1 #include <iostream>
2 #include <list>
3 #include <algorithm>
4
5 class MyList
6 {
7     std::list<int> l;
8     public:
9         MyList(int i = 0) : l(std::list<int>(i)) {}
10        void add(int i) { l.push_back(i); }
11        void print()
12        {
13            for(int i : l)
14                std::cout << i << " ";
15                std::cout << std::endl;
16        }
17    };
18
19 int main()
20 {
21     std::list<MyList<int>> list(5);
22     for(auto it = list.begin(), int n = 4; it != list.end(); it++, n--)
23     {
24         MyList<int> l;
25         for(int i = n; i >= 0; i--)
26             l.add(i);
27         *it = l;
28     }
29     std::sort(list.begin(), list.end()); //trie les listes par taille croissante
30     for(MyList<int> l : list)
31         l.print();
32 }
```

On ne peut pas comparer deux listes du coup il faut faire la surcharge d'opérateur de comparaison

Affichage souhaité du code 8

```

1 0
2 0 1
3 0 1 2
4 0 1 2 3
5 0 1 2 3 4

```

Affichage produit par le code 8

```
1 Erreur de compilation
```

Code 9

```

1 #include <iostream>
2
3 class A
4 {
5     int i;
6     public:
7         A(int i = 0) : i(i) {}
8         double operator +(double d) { return i + d; }
9         operator int() { return i; }
10        operator double() { return i; }
11    };
12
13 int main()
14 {
15     A a(2); std::cout << (a + 5) << std::endl;
16 }

```

Solution : changer
dans la surcharge
d'opérateur + le
paramètre
double en int

Affichage souhaité du code 9

```
1 7
```

Affichage produit par le code 9

```
1 Erreur de compilation
```

Nom :

Prénom :

Groupe :

Matricule :

Code 10

```
1 #include <iostream>
2
3 class A
4 {
5     protected:
6         int i;
7     public:
8         A(int i = 0) : i(i) {}
9         void print() { std::cout << i << std::endl; }
10    };
11
12 class B : public A
13 {
14     int j;
15     public:
16         B(int i = 0) : A(i + 2), j(i + 3) {}
17         void print() { std::cout << (i + j) << std::endl; }
18    };
19
20 int main() { A a = B(); a.print(); }
```

A prend uniquement les valeurs A instancié par le constructeur de A se trouvant dans celui de B

Solution : Pointeur de A qui pointe à l'adresse d'un nouvel objet B

Affichage souhaité du code 10

```
1 5
```

Affichage produit par le code 10

```
1 2
```

Code 11

```
1 #include <iostream>
2 class A
3 {
4     public:
5         B b;
6         void print() { std::cout << "A"; }
7     };
8 class B
9 {
10    public:
11        A a;
12        void print() { std::cout << "B"; }
13    };
14
15 int main() { A a; B b; a.print(); b.print(); }
```

Solution : supprimer la ligne 5, A ne connaît pas B, c++ lit de haut en bas.

Affichage souhaité du code 11

```
1 AB
```

Affichage produit par le code 11

```
1 Erreur de compilation
```

Nom :

Prénom :

Groupe :

Matricule :

Code 1

```
1 #include <iostream>
2 class A{
3     int * t;
4     int n;
5     public:
6     A(int * t, int n) : t(t), n(n) {}
7     ~A() { delete t; }
8     void print()
9     {
10         for(int i = 0; i < n; i++)
11             std::cout << t[i] << " ";
12     }
13 };
14
15 void f(A a) { std::cout << "GET_TO_THE_CHOPPAAA !!!" << std::endl; }
16
17 int main()
18 {
19     int t[5] = {1,2,3};
20     A a(t, 5);
21     f(a);
22     a.print(); std::cout << std::endl;
23
24     int s[4] = {4,5,6};
25     A b(s, 4);
26     b = a;
27     b.print(); std::cout << std::endl;
28
29     b = b;
30     b.print(); std::cout << std::endl;
31 }
```

rajouter explicitement deux
elements qui valent 0 dans le
tableau t

Affichage souhaité du Code 1

```
1 GET TO THE CHOPPAAA !!!
2 1 2 3 0 0
3 1 2 3 0 0
4 1 2 3 0 0
```

Nom :

Prénom :

Groupe :

Matricule :

Code 5

```
1 #include <iostream>
2 using namespace std;
3
4 class A
5 {
6     public:
7         virtual A& operator=(const A&) { cout << "Aff_A" << endl; }
8     };
9 class B : public A
10 {
11     public:
12         virtual B& operator=(const B&) { cout << "Aff_B" << endl; }
13     };
14
15 int main()
16 {
17     A * a1 = new A(); A * a2 = new A(); B * b1 = new B(); B * b2 = new B();
18     b1 = b2; a1 = b1; a1 = b2;
19     a1 = new A(); a2 = new A(); b1 = new B(); b2 = new B();
20     *b1 = *b2; *a1 = *b1; *a1 = *b2;
21 }
```

Faire des delete explicite
puisqu'on alloue
dynamiquement des objects
dans la memoire

Affichage souhaité du Code 5

```
1 Aff B
2 Aff A
3 Aff A
```

Code 6

```
1 #include <iostream>
2 #include <list>
3 using namespace std;
4
5 int main()
6 {
7     list<int> l = {1,2,3,4,5};
8     for(int i = 0; i < l.size(); i++)
9     {
10         auto j = l.rend(); j++;
11         cout << *j << " ";
12         l.pop_back();
13     }
14 }
```



Affichage souhaité du Code 6

```
1 5 4 3 2 1
```

Nom :

Prénom :

Groupe :

Matricule :

Code 7

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 class A
6 {
7     public:
8     ~A() { cout << "Boom" << endl; }
9 };
10
11 class B
12 {
13     vector<A*> v;
14     public :
15     B() : v(vector<A*>(4)) {}
16 };
17
18 void f() { B b; }
19
20 int main()
21 {
22     f();
23     cout << "I_want_you_in_my_room" << endl << "(outdated_song)" << endl;
24 }
```

Le destructeur détruit les objets à la fin, impossible d'afficher boom en premier.
Solution : créer 3 objets A et transformer le destructeur en constructeur pour afficher boom en premier.

Affichage souhaité du Code 7

```
1 Boom
2 Boom
3 Boom
4 Boom
5 I want you in my room
6 (outdated song)
```

Code 8

```
1 #include <iostream>
2 #using namespace std ;
3
4 int main()
5 {
6     int i = 5; int j = 2; double d = 1.5; //vous ne pouvez pas modifier cette ligne
7     cout << ( i / j + d) << endl;
8 }
```



Affichage souhaité du Code 8

```
1 4
```

Nom :

Prénom :

Groupe :

Matricule :

Code 10

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5 class A {
6     public:
7         int i;
8     };
9
10 int main()
11 {
12     vector<A> v = {5, 4, 3, 2, 1};
13     sort(v.begin(), v.end());
14     for(A a : v)
15         cout << a.i << " ";
16 }
```

Créer un constructeur de A qui recevra un entier pour instancier l'attribut i et donc remplir le vecteur par des nouveaux objets

Ajouter un opérateur de comparaison qui va gérer les comparaisons entre objets A puisque la méthode sort fait appelle à l'opérateur de comparaison afin de trier

Affichage souhaité du Code 10

```
1 2 3 4 5
```

Question 2. Expliquez en détail le mécanisme d'inférence de type (comment il est mis en œuvre, à quoi sert-il, les mots-clé associés). /5

Question 3. Qu'est-ce que le mécanisme de la liste d'initialisation et dans quels cas ne peut-on pas s'en passer ? /5

Question 4. Détaillez comment mettre en œuvre des conversions implicites définies par l'utilisateur dans les cas suivants : /8

- d'un type de base vers un autre type de base,
- d'un type de base vers une classe,
- d'une classe vers un type de base,
- d'une classe vers une autre classe.

Question 5. Quelle est l'utilité du mot-clé `virtual` lors de la déclaration de l'entête d'une classe telle que /12

```
1 class B : public virtual A { ... }
```

Dans quels cas ne peut-on éviter son utilisation ?

R. Absil (abs)

7 janvier 2016

Consignes générales

1. Notez votre nom, prénom, groupe et matricule sur chacune des feuilles que vous remettez à votre maître-assistant, y compris les brouillons.
2. Rédigez vos réponses en français correct : soignez votre orthographe et votre grammaire.
3. Justifiez chacune de vos réponses de façon complète et concise : une réponse incorrecte justifiée avec de bons arguments peut rapporter quelques points, alors qu'une bonne réponse non justifiée sera systématiquement comptée comme nulle.
4. L'examen est à cahier fermé.
5. L'examen dure 2h.

Question 1. Analysez les codes suivants, sachant qu'il n'y a pas d'erreur d'inclusion de fichier en-tête, et que toutes les options de compilation nécessaires sont fournies (p. ex., `-std=c++11`, `-fpermissive`, etc.). Pour chacun d'entre eux, indiquez

- s'il y a une erreur de compilation, à quelle(s) ligne(s) le cas échéant, et pourquoi;
- s'il y a une erreur lors de l'exécution, à quelle(s) ligne(s) le cas échéant, et pourquoi;
- si le programme s'exécute sans erreurs. Le cas échéant, indiquez les affichages qu'il génère, en spécifiant bien s'ils dépendent de l'architecture ou s'ils sont indéterminés.

Chaque code vaut 2 points. Répondez dans le tableau prévu à cet effet, situé à la page 7.

/30

Code 1

```
1 #include <iostream>
2 int f(int i = 4, double d) {return i * d; }
3 int main() {
4     std::cout << f(2.1) << std::endl;
5 }
```

ne compile pas, il faut initialiser D.
Attention à la conversion implicite des paramètres, Si je met 2.5 au premier paramètre et 3 au deuxième, il y aura une conversion implicite en double et int. Si je met 2.5 et 3.5, on aura une conversion implicite du int en double et on aura donc 2 doubles.
Solution : initialiser D.

Nom :

Prénom :

Groupe :

Matricule :

Code 2

```
1 #include <iostream>
2 #include <map>
3 int main() {
4     std::multimap<char, int> mci {{'a', 1}, {'b', 2}, {'c', 3},
5     {'a', 11}, {'a', 111}, {'b', 22}};
6     for (auto it = mci.cbegin(); it != mci.cend(); ++it)
7     {
8         ++it;
9         std::cout << it->first << ',' << it->second << std::endl;
10        ++it;
11    }
12 }
```

le code fonctionne mais il n'affiche pas toutes les paires.

Solution: enlever `++it;` et `+it;` après ça tout s'affiche dans l'ordre croissant des clefs, a puis b puis c

Code 3

```
1 #include <iostream>
2 int main() {
3     unsigned char c {'R'};
4     std::cout << c++ << std::endl;
5     signed float f {8.3};
6     std::cout << ++f << std::endl;
7 }
```

Le code ne compile pas car float ne peut pas être un signe
Solution : enlever le signed

Code 4

```
1 #include <iostream>
2 class A {
3     public:
4         void f(int) { std::cout << "A::f(int)" << std::endl; }
5     };
6 class B : public A {
7     public:
8         void f(double) { std::cout << "B::f(double)" << std::endl; }
9     };
10 int main() {
11     B b;
12     b.f(5);
13 }
```

X

Code 5

```
1 #include <iostream>
2 int main()
3 {
4     int i = 33;
5     int const & ri = i;
6     std::cout << ri << std::endl;
7     i++;
8     std::cout << ri << std::endl;
9 }
```

X

Code 6

```

1 #include <iostream>
2 int main()
3 {
4     const int i = 33;    int const * pti = &i;
5     std::cout << *pti << std::endl;
6     (*pti)++;
7     std::cout << *pti << std::endl;
8 }
```

le code ne compile pas
Solution : (*pti)++ • -> pti++

Code 7

```

1 #include <iostream>
2 int f() {
3     static int i { 4 };
4     return --i;
5 }
6 int main() {
7     std::cout << f() << std::endl;
8     while (f()) { }
9     std::cout << f() << std::endl;
10 }
```

le code est ok et affiche 3 et
-1, si on change en post
incrémentation il affichera 4
et -1

Code 8

```

1 #include <iostream>
2 class A {
3     public:
4     A() { std::cout << "+A" << std::endl; }
5     ~A() { std::cout << "-A" << std::endl; }
6 };
7 int main() {
8     A * a = new A();
9     new A;
10 }
```

le code est ok mais il n'affiche que deux
fois "+A" car quand on fait
des new, on est obligé d'appeler
explicitement le constructeur pour
détruire l'objet.
Solution : appeler 2 fois le destructeur
explicitement • delete a

Code 9

```

1 #include <iostream>
2 void f(int) { std::cout << "int" << std::endl; }
3 void f(double) { std::cout << "double" << std::endl; }
4 int main() {
5     f(5.);
6     f('5');
7     f(5u);
8 }
```

le code ne compile pas, car f ne connaît
pas 5u à la ligne 7.
Solution : surdéfinir f pour qu'il puisse
connaître 5u

Code 10

```

1 #include <iostream>
2 #include <cmath>
3
4 int main() {
5     std::cout << (sqrt(-1) == sqrt(-1)) << std::endl; // il y a deux = entre les sqrt...
6 }
```

le code est ok mais il affiche false
sqrt(-1) n'existe pas car racine carré d'un
négatif n'est pas possible
donc ça renvoie NaN et n'est Nan n'est pas
comparable, on ne peut
pas comparer 2 NaN donc il renvoie false.

Nom :

Prénom :

Groupe :

Matricule :

Code 11

```
1 #include <iostream>
2 using namespace std;
3 class A{
4     public:
5         A() { cout << "+A" << endl; }
6         A(const A& a) { cout << "rA" << endl; }
7         virtual ~A() { cout << "-A" << endl; }
8     };
9 class B{
10    public:
11        B() { cout << "+B" << endl; }
12        B(const B& b) { cout << "rB" << endl; }
13        virtual ~B() { cout << "-B" << endl; }
14    };
15 class C : public A, public virtual B{
16    public:
17        C() { cout << "+C" << endl; }
18        C(const C& c) { cout << "rC" << endl; }
19        virtual ~C() { cout << "-C" << endl; }
20    };
21 void f(A a) {}
22
23 int main()
24 {
25     C c; cout << endl;
26     f(c); cout << endl;
27     C * cc = new C(); cout << endl;
28     delete cc; cout << endl;
29 }
```

Code 12

```
1 #include <iostream>
2 class A
3 {
4     public:
5         virtual A& operator =(const A&) { std::cout << "=A" << std::endl; }
6     };
7 class B : public A
8 {
9     public:
10        virtual B& operator =(const B&) { std::cout << "=B" << std::endl; }
11    };
12 int main()
13 {
14     A * a1 = new A; A * a2 = new A; B * b1 = new B;
15     a1 = a2; a1 = b1;
16     std::cout << "done" << std::endl;
17 }
```

le code est ok mais il n'affiche pas les A=

Solution : pour afficher les A=, changer a1 = a2 en *a1 = *a2

Comme c'est B est la classe dérivé et A la classe de base

a1 peut être = à b1 mais b1 ne peut pas être = à a1

Code 13

```
1 #include <iostream>
2 int main()
3 { int i = 5; int j = 2; double d = 1.5;
4   cout << (i / j + d) << endl;
5 }
```

le code est ok mais affichera 3.5 au lieu du 4 car le résultat de i/j vaudra 2 en entier et non en double, il faudra faire donc un cast

Nom :

Prénom :

Groupe :

Matricule :

Code 14

```
1 #include <iostream>
2 class A{
3     int * t;
4     int n;
5     public:
6         A(int * t, int n) : t(t), n(n) {}
7         ~A() { delete t; }
8         void print()
9         {
10             for(int i = 0; i < n; i++)
11                 std::cout << t[i] << " ";
12         }
13     };
14 void f(A a) { std::cout << "f" << std::endl; }
15 int main()
16 {
17     int t[5] = {1,2,3};
18     A a(t, 5);
19     a.print(); std::cout << std::endl;
20     f(a);
21     a.print();
22 }
```

Code 15

```
1 #include <iostream>
2 class CA {
3     private :
4         int i_ = 3;
5     public :
6         CA(int i_) : i_{ i_ } {}
7         int i() const { return i_ ; }
8     };
9 int main () {
10     std::cout << CA().i() << std::endl;
11 }
```

le code ne compile pas car il n'y a pas d'objet CA à afficher
Solution : créer un objet CA et l'afficher

Répondez sur le papier ministre pour les questions suivantes

Question 2. Quelles sont les différences notables entre les références et les pointeurs ?
Détaillez ces différences entre autre du point de vue de leur manipulation, de la gestion mémoire, de ce qui est permis avec un concept et pas avec l'autre, etc. /5

Question 3. Quelle est la différence entre la redéfinition et la surdéfinition de fonction / méthode en C++ ? Quelles sont les règles d'appel de fonctions dans ces deux cas ? /5



Langage C/C++ - Examen

R. Absil (abs), N. Vansteenkiste (nvs), M. Wahid (mwa)

/100

20 mai 2015

Consignes générales

1. Notez votre nom, prénom et groupe sur chacune des feuilles que vous remettez à votre maître-assistant, y compris les brouillons.
2. Rédigez vos réponses en français correct : soignez votre orthographe et votre grammaire.
3. Lisez les consignes et énoncés attentivement pour vous assurer de bien cerner les questions.
4. Justifiez chacune de vos réponses de façon complète et concise : une réponse incorrecte justifiée avec de bons arguments peut rapporter quelques points, alors qu'une bonne réponse non justifiée sera systématiquement comptée comme nulle.

Rappel : le langage de référence est le C++11 (ISO/IEC 14882 :2011) limité à son support par gcc 4.8.0.

Question 1. Analysez les codes suivants, sachant qu'il n'y a pas d'erreur d'inclusion de fichier en-tête, et que toutes les options de compilation nécessaires sont fournies (p. ex., `-std=c++11`, `-fpermissive`, etc.). Pour chacun d'eux, indiquez

/40

- s'il y a une erreur de compilation, à quelle(s) ligne(s) le cas échéant, et pourquoi ;
- s'il y a une erreur lors de l'exécution, à quelle(s) ligne(s) le cas échéant, et pourquoi ;
- si le programme s'exécute sans erreurs. Le cas échéant, indiquez les affichages qu'il génère, en spécifiant bien s'ils dépendent de l'architecture ou s'ils sont indéterminés.

Chaque code vaut deux points. Répondez dans le tableau prévu à cet effet, commençant à la page 7.

Code 1	Note
<pre> 1 #include <iostream> 2 int f(const char tab []) { return sizeof(tab) / sizeof(*tab); } 3 int main() 4 { 5 char bat [] = {1, 2, 3, 4, 5, 6, 7}; 6 std::cout << sizeof(bat) / sizeof(*bat) << std::endl; 7 std::cout << f(bat) << std::endl; 8 }</pre>	<p>il fera uniquement le sizeof du premier élément du tableau et pas de l'entier du tableau du coup ça retournera 1 car 4 bytes / 4 bytes</p>

Code 2	Note
<pre> 1 #include <iostream> 2 int f(int i = 3) { 3 i *= 2; 4 return i; 5 } 6 int main() { 7 std::cout << f() << std::endl; 8 std::cout << ++f() << std::endl; 9 }</pre>	On ne peut pas incrementer une fonction comme ce n'est pas une variable entière
Code 3	Note
<pre> 1 #include <iostream> 2 class A{ 3 public: 4 int& i; 5 A(int j) { i = j; } 6 }; 7 int main(){ 8 std::cout << A(3).i << std::endl; 9 }</pre>	Enlever la référence de i
Code 4	Note
<pre> 1 #include <iostream> 2 const char * f(const int tab[]) { 3 return "bop"; 4 } 5 int main() { 6 unsigned bat [] {1, 2, 3, 4, 5, 6, 7}; 7 std::cout << "bip"; 8 std::cout << f(bat) << std::endl; 9 }</pre>	remplacer unsigned bat par const int bat
Code 5	Note
<pre> 1 #include <iostream> 2 class CA { 3 private: 4 int i_ = -3; 5 public: 6 CA(int i) : i_(i) {} 7 int i() const { return i_; } 8 }; 9 int main() { 10 std::cout << CA().i() << std::endl; 11 }</pre>	X
Code 6	Note
<pre> 1 #include <iostream> 2 #include <vector> 3 #include <algorithm> 4 int main() { 5 std::vector<char> vc {'i', 'e', 'a', 'u', 'y', 'o'}; 6 std::sort(vc.begin() + 1, vc.end() - 1); 7 for (auto e : vc) { 8 std::cout << e << " "; 9 } 10 std::cout << std::endl; 11 }</pre>	X

Code 7	Note
<pre> 1 #include <iostream> 2 int main(){ 3 auto i; i = 2; 4 std :: cout << i << std :: endl; 5 i = 2.1; 6 std :: cout << i << std :: endl; 7 }</pre>	auto doit être assigner pour définir le type à choisir
Code 8	Note
<pre> 1 #include <iostream> 2 #include <list> 3 int main(){ 4 std :: list<int> s; 5 for(int i = 0; i <= 6; i++) 6 s.push_back(i * i - i); 7 for(int i = 0; i < s.size(); i++) 8 { 9 auto j = s.rend(); j++; 10 std :: cout << *j << "\u00a0"; 11 s.pop_back(); 12 } 13 }</pre>	X
Code 9	Note
<pre> 1 #include <iostream> 2 class A { 3 public: 4 void f(int n) 5 { std :: cout << "A::entier\u00a0" << n << std :: endl; } 6 7 void f(char n) 8 { std :: cout << "char\u00a0" << n << std :: endl; } 9 }; 10 class B : public A { 11 public: 12 void f(int n, int m) 13 { std :: cout << "entiers\u00a0" << n << "\u00a0et\u00a0" << m << std :: endl; } 14 }; 15 int main(){ 16 int n = 1; char c = 'a'; B b; 17 b.f(n); 18 b.f(c); 19 }</pre>	X
Code 10	Note
<pre> 1 #include <iostream> 2 #include <set> 3 int main() { 4 std :: multiset<int> msi = {3, 5, 3, 6, 2}; 5 for (auto e : msi) { 6 std :: cout << e << "\u00a0"; 7 } 8 std :: cout << std :: endl; 9 }</pre>	X
Code 11	Note
<pre> 1 #include <iostream> 2 #include <utility> 3 int main() { 4 std :: pair<char, double> pcd('Z', -3.14); 5 std :: pair<float, long> pfl = pcd; 6 std :: cout << pcd.first << "\u00a0" << pfl.second 7 }</pre>	On fait une copie de pcd et comme une conversion implicite se produit d'un double en long, on affiche -3 et pas -3.14

Nom :

Prénom :

Matricule :

Groupe :

Code 18	Note
<pre> 1 #include <iostream> 2 3 class point; 4 class circle 5 { 6 point c; 7 double radius; 8 9 public : 10 circle(point p, double rad) : c(p), radius(rad) {} 11 12 void translate(double x, double y); 13 void print(); 14}; 15 16 class point { 17 int x, y; 18 public : 19 point(double abs=0, double ord=0) : x(abs), y(ord) {} 20 21 friend void circle::translate(double x, double y); 22 friend void circle::print(); 23}; 24 25 void circle::translate(double x, double y) { 26 c.x += x; 27 c.y += y; 28 } 29 30 void circle::print(){ 31 std::cout << c.x << " " << c.y << " " << radius << std::endl; 32 } 33 34 int main() { 35 point p(1,1); 36 circle c(p, 2); 37 c.translate(1,-1); 38 c.print(); 39 }</pre>	
Code 19	Note
<pre> 1 #include <iostream> 2 3 int main() { 4 int i = 2.5; 5 cout << i << endl; 6 }</pre>	Il manque le std dans cout et endl
Code 20	Note
<pre> 1 #include <iostream> 2 int a = 5; 3 int main() 4 { 5 if (!(a % 2)) 6 a--; 7 else{ 8 std::cout << a << " " << std::endl; 9 a++; 10 } 11 main(); 12 }</pre>	Récursivité infini comme le main rentre à chaque fois dans le main sans condition d'arrêt

Matricule :

Nom :

Prénom :

Groupe :

Code 1

```
1 #include <iostream>
2 #include <string>
3
4 const char* hello()
5 {
6     const char str [] = "Hello there !";    il faut remplacer par char *
7     return str;
8 }
9
10 int main()
11 {
12     std::string s(hello());
13     std::cout << s << std::endl;
14 }
```

Affichage souhaité du Code 1

```
1 Hello there !
```

Affichage produit par le Code 1

```
1 Erreur de segmentation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 2

```
1 #include <stdio.h>
2
3 int min(const int tab[], int n)
4 {
5     int min;
6     for(int i = 0; i < n; i++)
7         if(min > tab[i])
8             min = tab[i];
9
10    return min;
11 }
12
13 int main()
14 {
15     const int tab[] = {4, 3, 1, 8, 5, 9, 10};
16     printf("%d\n", min(tab, 7));
17 }
```

Il n'y a pas de valeur par défaut en C. Aussi,
sans initialisation ou affectation, la valeur
d'une variable est indéterminée ! Veillez donc
à ce que vos variables aient une valeur connue
avant de les utiliser !

Affichage souhaité du Code 2

```
1 1
```

Affichage produit par le Code 2

```
1 -42
```

Matricule :

Nom :

Prénom :

Groupe :

Code 3

```
1 #include <iostream>
2 #include <list>
3
4 int main()
{
    std::list<int> l;
    for(int i = 0; i <= 6; i++)
        l.push_back(i * i - i);
    for(int i = 0; i < l.size(); i++)
    {
        std::cout << l.back() << " ";
        l.pop_back();
    }
}
```

Comme on supprime les éléments de la liste après chaque pop back, la taille de la liste se retrouve affecter faisant en sorte de réduire le nombre de boucles i s'arrêtera beaucoup plus tôt
Il faut créer une variable qui stockera la taille de la liste et qui se verra inchangé

Affichage souhaité du Code 3

```
1 30 20 12 6 2 0 0
```

Affichage produit par le Code 3

```
1 30 20 12 6
```

Matricule :

Nom :

Prénom :

Groupe :

Code 4

```
1 #include <stdio.h>
2
3 void swap(int * i, int * j)
4 {
5     int * tmp = i;
6     i = j; j = tmp;
7 }
8
9 int main()
10{
11    int i = 2; int j = 3; swap(&i, &j);
12    printf("%d %d\n", i, j);
13}
```

On swappe les adresses qui sont contenues dans les copies de pointeurs

Placer des * devant chaque variable afin de changer la valeur qui est directement pointé à leurs adresses.

Affichage souhaité du Code 4

```
1 3 2
```

Affichage produit par le Code 4

```
1 2 3
```

Code 5

```
1 #include <stdio.h>
2
3 int* give_me_a_one()
4 {
5     int * pt = malloc(sizeof(int));
6     int * pt = 1; Ne pas déclarer mais instancier le pt à 1 après allocation
7     return pt;           dynamique
8
9 int main()
10{
11    printf("%d\n", *give_me_one());           int * one = give_me_a_one();
12}                                         printf("%d\n", *one);
13                                         free(one);
```

Affichage souhaité du Code 5

```
1 1
```

Affichage produit par le Code 5

```
1 Erreur de segmentation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 6

```
1 #include <stdio.h>
2
3 int* give_me_a_one()
4 {
5     int * pt;           Faire un malloc de pt , il est obligatoire d'instancier le pointeur pour qu'il
6     *pt = 1;            puisse pointer vers une référence et donc accéder à la valeur
7 }
8
9 int main()
10 {
11     printf("%d\n", *give_me_one());
12 }
```

Affichage souhaité du Code 6

1 1

Affichage produit par le Code 6

1 Erreur de segmentation

Code 7

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a1 = 2.1;           conversion implicite de double à int sauf pour la dernière
7     cout << a1 << " ";
8     int a2(2.1);           déclaration qui nécessite de faire une conversion
9     cout << a2 << " ";   explicite
10    int a3{2.1};           int a2{(int)2.1};
11    cout << a3 << endl;   ou
12 }
```

Affichage souhaité du Code 7

1 2 2 2

Affichage produit par le Code 7

1 Erreur de compilation

Code 8

```

1 #include <iostream>
2
3 struct A
4 {
5     int i;
6     A(int i) { this->i = i; }
7 }; A(A &a){ this->i = a.i; };
8
9 struct B
10 {
11     A a;
12     B(A a) { this->a = a; } Liste d'initialisation à placer entre la mise en paramètre et les accolades (le corps du constructeur)
13 }; B(A a):a(a){};
14
15 int main()
16 {
17     A a(2); B b(a);
18     std::cout << b.a.i << std::endl;
19 }
```

Affichage souhaité du Code 8

1 2

Affichage produit par le Code 8

1 Erreur de compilation

Matricule :

Nom :

Prénom :

Groupe :

Code 10

```
1 #include <stdio.h>
2
3 void print_reverse(int tab[], unsigned size)
4 {
5     for(unsigned i = size - 1; i >= 0; i--)
6         printf("%d ", tab[i]);
7     printf("\n");
8 }
9
10 int main()
11 {
12     int tab[] = {1, 2, 3, 4, 5, 6, 7, 8};
13     reverse_print(tab, 8);
14 }
```

Comme la variable size est un entier unsigned, il est strictement positif, il ne valera jamais une valeur négatif

ça fera que la boucle ne s'arrête jamais, puisque i aura comme valeur, la valeur maximal d'un entier unsigned

Affichage souhaité du Code 10

```
1 8 7 6 5 4 3 2 1
```

Affichage produit par le Code 10

```
1 Boucle infinie
```

Code 11

FICHIER B.CPP

```

1 #include <iostream>
2
3 class A;
4 class B
{
5     public:
6         A * b;
7         B() : b(nullptr) {}
8         void print() { std::cout << "B" << std::endl; }
9
10};

```

FICHIER A.CPP

```

1 #include <iostream>
2
3 #include "B.cpp"
4
5 class A
{
6     public:
7         B b;
8         A() {}
9         void print() { std::cout << "A" << std::endl; }
10
11};

```

FICHIER MAIN.CPP

```

1 #include <iostream>
2 #include "B.cpp"           Multiples inclusions, il faut retirer celui-là
3 #include "A.cpp"
4
5 int main()
6 {
7     B b; A a;           Ajout possible dans un header pour
8     b.a = &a; a.b = b;   corriger
9     b.print(); a.print();    ifdef
10}

```

Affichage souhaité du Code 11

1 B
2 A

Affichage produit par le code 11

1 Erreur de compilation

Matricule :

Nom :

Prénom :

Groupe :

Code 12

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void make_tab(int* tab, int size)
5 {
6     tab = (int*) malloc(size * sizeof(int));
7     for(int i = 0; i < size; i++)
8         tab[i] = i;
9 }
10
11 int main()
12 {
13     int * tab = NULL;
14     int * tab = (int*) malloc(5 * sizeof(int));
15     make_tab(tab, 5);
16
17     for(int i = 0; i < 5; i++)
18         printf("%d ", tab[i]);
19     printf("\n");
20 }
```

on fait une allocation mémoire sur une copie de pointeur de tab

alors qu'on est sensé le faire pour le tab se trouvant dans le main

Affichage souhaité du Code 12

```
1 0 1 2 3 4
```

Affichage produit par le Code 12

```
1 Erreur de segmentation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 1

```
1 #include <iostream>
2 using namespace std;
3
4 class exceptA {};
5 class exceptB : public exceptA {};
6 class exceptC : public exceptB {};
7
8 void f() { throw exceptB(); }
9
10 int main()
11 {
12     try
13     {
14         f();
15     }
16     catch(exceptA& e)
17     {
18         cout << "I caught an A" << endl;
19     }
20     catch(exceptB& e)
21     {
22         cout << "I caught a B" << endl;
23     }
24     catch(exceptC& e)
25     {
26         cout << "I caught a C" << endl;
27     }
28     catch(...)
29     {
30         cout << "I caught something" << endl;
31     }
32 }
```

Affichage souhaité du Code 1

1 I caught a B

Affichage produit par le code 1

1 I caught an A

Matricule :

Nom :

Prénom :

Groupe :

Code 2

```
1 #include <iostream>
2 #include <vector>
3 struct A {
4     int i;
5     A(int i) : i(i) {}
6     void print() { std::cout << i << "\u00b7"; }
7 };
8 int main() {
9     std::vector<A> v(5);
10    for (int j = 0; j < v.size(); j++)
11        v[j].i = j;
12    for (A a : v)
13        a.print();
14 }
```

Affichage souhaité du Code 2

1 0 1 2 3 4

Affichage produit par le code 2

1 Erreur de compilation

Code 3

```
1 #include <iostream>
2 struct A {
3     virtual A& operator=(const A&) { std::cout << "A" << "\u00b7"; }
4 struct B : A {
5     virtual B& operator=(const B&) { std::cout << "B" << "\u00b7"; }
6 int main() {
7     A * a1 = new A; A * a2 = new A; B * b1 = new B; B * b2 = new B;
8     *a1 = *a2; *b1 = *b2; *a1 = *b1;
9 }
```

Affichage souhaité du Code 3

1 A B B

Affichage produit par le code 3

1 A B A

Matricule :

Nom :

Prénom :

Groupe :

Code 4

```
1 #include <iostream>
2
3 struct A
4 {
5     int * a;
6     A(int i = 0) : a(new int(i)) {}
7     ~A() { delete a; }
8     void print() { cout << *a << " "; }
9 };
10
11 void f(A a) { }
12
13 int main()
14 {
15     A a(2);
16     f(a);
17     a.print();
18
19     A aa;
20     aa = a;
21     aa.print();
22
23     aa = aa;
24     aa.print();
25 }
```

Affichage souhaité du Code 4

```
1 2 2 2
```

Affichage produit par le Code 4

```
1 Erreur de segmentation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 5

```
1 #include <iostream>
2 template<class T> struct Wrapper {
3     T t; int i;
4     Wrapper(T t) : t(t) {}  
5     Wrapper(int i) : i(i) {}
6     void print() { std::cout << t << " " << i << std::endl; }
7 };
8 int main() {
9     Wrapper<int> b(2); b.t = 3;
10    b.print();
11 }
```

Affichage souhaité du Code 5

1 3 2

Affichage produit par le Code 5

1 Erreur de compilation

Code 6

```
1 #include <iostream>
2 template<int N> struct Number {
3     int n;
4     Number() : n(N) { std::cout << n << std::endl; }
5 };
6 int fib(int n) {
7     if(n == 0 || n == 1) return 1;
8     else return fib(n-1) + fib(n-2);
9 }
10 int main() {
11     Number<fib(5)> n;
12 }
```

Affichage souhaité du Code 6

1 8

Affichage produit par le Code 6

1 Erreur de compilation

Matricule :

Nom :

Prénom :

Groupe :

Code 7

```
1 #include <iostream>
2 #include <list>
3 template<class T> struct MyList {
4     std::list<T>& v;
5     MyList(std::list<T>& v) : v(v) {}
6 };
7 template<class Container, class Fct> void forall(Container c, Fct f) {
8     for(auto e : c)
9         f(e);
10 }
11 int main() {
12     std::list<int> l = {0, 1, 2, 3, 4, 5};
13     MyList<int> myl(l);
14     forall(myl, [](int i) { std::cout << i << " " });
15 }
```

Affichage souhaité du Code 7

1 0 1 2 3 4 5

Affichage produit par le Code 7

1 Erreur de compilation

Code 8

```
1 #include <iostream>
2
3 double operator "" ms(long double x) { return 0.001 * x; }
4 double operator "" s(long double x) { return x; }
5
6 int main() {
7     std::cout << "1ms is " << 1ms << " s" << std::endl;
8 }
```

Affichage souhaité du Code 8

1 1ms is 0.001 s

Affichage produit par le Code 8

1 Erreur de compilation

Matricule :

Nom :

Prénom :

Groupe :

Code 9

```
1 #include <iostream>
2 #include <vector>
3 class MyVector
4 {
5     std::vector<int> v;
6 public:
7     MyVector(int i = 0) : v(std::vector<int>(i)) {}
8     void add(int i) { v.push_back(i); }
9     void print()
10    {
11        for(int i : v)
12            std::cout << i << " ";
13        std::cout << std::endl;
14    }
15};
16
17 int main()
18{
19    std::vector<MyVector> vector(5); int n = 4;
20    for(auto it = vector.begin(); it != vector.end(); it++)
21    {
22        MyVector v;
23        for(int i = n; i >= 0; i--)
24            v.add(i);
25        *it = v; n--;
26    }
27    std::sort(vector.begin(), vector.end()); //trie les vector par taille croissante
28    for(MyVector v : vector)
29        v.print();
30}
```

Affichage souhaité du Code 9

```
1 0
2 0 1
3 0 1 2
4 0 1 2 3
5 0 1 2 3 4
```

Affichage produit par le Code 9

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 10

```
1 #include <iostream>
2 using namespace std;
3
4 struct Mere
5 {
6     void f(int n) {cout << "Mere::integer" << n << endl; }
7     void f(char n) {cout << "Mere::character" << n << endl; }
8 };
9
10 struct Fille : public Mere
11 {
12     void f(int n, int m)
13     {
14         cout << "Fille::integers" << n << " " << m << endl;
15     }
16 };
17
18 int main()
19 {
20     int n = 1;
21     char c = 'a';
22     Fille b;
23     b.f(n);
24     b.f(c);
25     b.f(n, c);
26 }
```

Affichage souhaité du Code 10

```
1 Mere::integer
2 Mere::character
3 Fille::integers
```

Affichage produit par le Code 10

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 11

```
1 #include <iostream>
2 using namespace std;
3
4 struct Poulet
5 {
6     Poulet() { cout << "+_"; }
7     Poulet(const A& a) { cout << "r_"; }
8     ~Poulet() { cout << "-_"; }
9     Poulet& operator =(const Poulet& a)
10    {
11        cout << "=_" << endl;
12        return *this;
13    }
14};
15
16 void f(Poulet a) {}
17 void g(Poulet& a) {}
18
19 int main()
20 {
21     Poulet a;
22     f(a);
23     Poulet * aa = new Poulet();
24     aa = &a;
25     g(*aa);
26 }
```



Affichage souhaité du Code 11

```
1 + r - + = - -
```

Affichage produit par le Code 11

```
1 + r - + -
```

Matricule :

Nom :

Prénom :

Groupe :

Code 12

```
1 #include <iostream>
2 class A
3 {
4     protected:
5         int i;
6     public:
7         A(int i = 0) : i(i) {}
8         void print() { std::cout << i << std::endl; }
9     };
10 class B : public A
11 {
12     int j;
13     public:
14         B(int i = 0) : A(i + 2), j(i + 3) {}
15         void print() { std::cout << (i + j) << std::endl; }
16     };
17 int main() { A a = B(); a.print(); }
```

Affichage souhaité du Code 12

1 5

Affichage produit par le Code 12

1 2

Code 13

```
1 #include <iostream>
2 struct A { void print() { std::cout << "A" << std::endl; } }
3 struct B : A { void print() { std::cout << "B" << std::endl; } }
4 int main() {
5     B b; A& a = b;
6     a.print();
7 }
```

Affichage souhaité du Code 13

1 B

Affichage produit par le code 13

1 A

Matricule :

Nom :

Prénom :

Groupe :

Code 14

```
1 #include <iostream>
2
3 template<class T = int> class A
4 {
5     T t;
6
7     public:
8         A(const T& t = T()) : t(t) {}
9         T& operator() () { return t; }
10    };
11
12 int main()
13 {
14     A<int> i(2);
15     A<double> f(2.2);
16     std::cout << i() << " " << f() << std::endl;
17
18     i = f;
19     std::cout << i() << " " << f() << std::endl;
20 }
```

Affichage souhaité du Code 14

```
1 2 2.2
2 2
```

Affichage produit par le code 14

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 15

```
1 struct Duck { void quack() {} };
2 struct Pig { void groink() {} };
3
4 template<class T> auto test_duck(const T& t, int)
5     -> decltype(t.quack(), bool())
6 { return true; }
7
8 template<class T> auto test_duck(const T& t, long)
9 { return false; }
10
11 template<class T> bool is_duck(const T& t)
12 {
13     return test_duck(t, 0);
14 }
15
16
17 int main()
18 {
19     Duck duck; Pig pig;
20
21     cout << is_duck(duck) << " ";
22     cout << is_duck(pig) << endl;
23 }
```

Affichage souhaité du Code 15

```
1 0
```

Affichage produit par le code 15

```
1 0 0
```

Matricule :

Nom :

Prénom :

Groupe :

Code 16

```
1 #include <iostream>
2 #include <type_traits>
3
4 template<class T>
5 using isNotInt = std::enable_if_t<! std::is_same_v<T, int>>;
6
7 template<class T>
8 class A
9 {
10     T t;
11     int i;
12
13 public:
14     A(int i) : i(i) { cout << i << "\n"; }
15
16     template<class = isNotInt<T>>
17     A(T t) : t(t), i(-1) { cout << i << "\n"; }
18 };
19
20 int main()
21 {
22     A<double> a(1.1); //ok
23     A<int> b(1); //ok
24 }
```

Affichage souhaité du Code 16

```
1 1.1 1
```

Affichage produit par le code 16

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 17

```
1 #include <iostream>
2 #include <string>
3 #include <type_traits>
4
5 struct Duck
6 {
7     std::string name;
8     Duck(std::string name) : name(name) {}
9     void quack() const { cout << name << ":~quack~!"; }
10 };
11
12 template<class>
13 struct sfinae_true : std::true_type {};
14
15 template<class T>
16 static auto test_duck(int)
17     -> sfinae_true<decltype(T().quack())>;
18
19 template<class>
20 static auto test_duck(long)
21     -> false_type;
22
23 template<class T>
24 struct bool_duck : decltype(test_duck<T>(0)) {};
25
26 int main()
27 {
28     cout << (bool_duck<Duck>() == true) << endl;
29 }
```

Affichage souhaité du Code 17

1

Affichage produit par le Code 17

0

Matricule :

Nom :

Prénom :

Groupe :

Code 18

```
1 #include <iostream>
2
3 struct A { void a() { std::cout << "A" << std::endl; } };
4
5 struct B {
6     int;
7     void b() { cout << "B" << endl; }
8 };
9
10 template<class T> bool isA(T t) { return sizeof(t) == sizeof(A); }
11
12 template<class T> bool isB(T t) { return sizeof(t) == sizeof(B); }
13
14 template<class T> void f(T t)
15 {
16     if (isA(t))
17         t.a();
18     else
19         t.b();
20 }
21
22 int main()
23 {
24     A a; B b;
25     f(a); f(b);
26 }
```

Affichage souhaité du Code 18

1

Affichage produit par le Code 18

0

Matricule :

Nom :

Prénom :

Groupe :

Code 19

```
1 #include <iostream>
2 using namespace std;
3 struct A {
4     A() { cout << "+A" << endl; }
5     A(const A& a) { cout << "rA" << endl; }
6     virtual ~A() { cout << "-A" << endl; }
7 };
8 struct B {
9     B() { cout << "+B" << endl; }
10    B(const B& b) { cout << "rB" << endl; }
11    virtual ~B() { cout << "-B" << endl; }
12 };
13 struct C : public A, public virtual B {
14     C() { cout << "+C" << endl; }
15     C(const C& c) { cout << "rC" << endl; }
16     virtual ~C() { cout << "-C" << endl; }
17 };
18
19 void f(A a) {}
20
21 int main() {
22     C c; cout << endl;
23     f(c); cout << endl;
24 }
```

Affichage souhaité du Code 19

```
1 +B    rA    -C    -C
2 +A        -A    -A
3 +C    rB    -B    -B
```

Affichage produit par le Code 19

```
1 +A    rA    -C    -C
2 +B    rB    -B    -B
3 +C    rC    -A    -A
```

Remarque 1. Les affichages ci-dessus sont à lire en colonnes, de haut en bas et de gauche à droite. Les lignes vides sont à ignorer.

Matricule :

Nom :

Prénom :

Groupe :

Code 20

```
1 #include <iostream>
2 #include <vector>
3 struct A {
4     int i;
5     A(int i) : i(i) {}
6     void print() { std::cout << i << "\u033"; }
7 };
8
9 struct B
10 {
11     A a;
12     B(A a){ this->a = a; }
13     void print() { std::cout << a.i << "\u033"; }
14 }
15
16 int main()
17 {
18     A a(1); B b(a);
19     b.print();
}
```

Affichage souhaité du Code 20

1 1

Affichage produit par le Code 20

1 Erreur de compilation

Matricule :

Nom :

Prénom :

Groupe :

Code 1

```
1 #include <iostream>
2 using namespace std;
3
4 class exceptA {};
5 class exceptB : public exceptA {};
6 class exceptC : public exceptB {};
7
8 void f() { throw exceptB(); }
9
10 int main()
11 {
12     try
13     {
14         f();
15     }
16     catch(exceptA& e)
17     {
18         cout << "I caught an A" << endl;
19     }
20     catch(exceptB& e)
21     {
22         cout << "I caught a B" << endl;
23     }
24     catch(exceptC& e)
25     {
26         cout << "I caught a C" << endl;
27     }
28     catch(...)
29     {
30         cout << "I caught something" << endl;
31     }
32 }
```



Affichage souhaité du Code 1

1 I caught a B

Affichage produit par le code 1

1 I caught an A

Matricule :

Nom :

Prénom :

Groupe :

Code 2

```
1 #include <iostream>
2 #include <vector>
3 struct A {
4     int i;
5     A(int i) : i(i) {}
6     void print() { std::cout << i << "\u00b7"; }
7 };
8 int main() {
9     std::vector<A> v(5);
10    for (int j = 0; j < v.size(); j++)
11        v[j].i = j;
12    for (A a : v)
13        a.print();
14 }
```



Affichage souhaité du Code 2

1 0 1 2 3 4

Affichage produit par le code 2

1 Erreur de compilation

Code 3

```
1 #include <iostream>
2 struct A {
3     virtual A& operator=(const A&) { std::cout << "A" << "\u00b7"; }
4 struct B : A {
5     virtual B& operator=(const B&) { std::cout << "B" << "\u00b7"; }
6 int main() {
7     A * a1 = new A; A * a2 = new A; B * b1 = new B; B * b2 = new B;
8     *a1 = *a2; *b1 = *b2; *a1 = *b1;
9 }
```

Affichage souhaité du Code 3

1 A B B

Affichage produit par le code 3

1 A B A

Matricule :

Nom :

Prénom :

Groupe :

Code 4

```
1 #include <iostream>
2
3 struct A
4 {
5     int * a;
6     A(int i = 0) : a(new int(i)) {}
7     ~A() { delete a; }
8     void print() { cout << *a << " " ; }
9 };
10
11 void f(A a) { }
12
13 int main()
14 {
15     A a(2);
16     f(a);
17     a.print();
18
19     A aa;
20     aa = a;
21     aa.print();
22
23     aa = aa;
24     aa.print();
25 }
```



Affichage souhaité du Code 4

```
1 2 2 2
```

Affichage produit par le Code 4

```
1 Erreur de segmentation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 5

```
1 #include <iostream>
2 template<class T> struct Wrapper {
3     T t; int i;
4     Wrapper(T t) : t(t) { }
5     Wrapper(int i) : i(i) { }
6     void print() { std::cout << t << " " << i << std::endl; }
7 };
8 int main() {
9     Wrapper<int> b(2); b.t = 3;
10    b.print();
```



Affichage souhaité du Code 5

```
1 3 2
```

Affichage produit par le Code 5

```
1 Erreur de compilation
```

Code 6

```
1 #include <iostream>
2 template<int N> struct Number {
3     int n;
4     Number() : n(N) { std::cout << n << std::endl; }
5 };
6 int fib(int n) {
7     if(n == 0 || n == 1) return 1;
8     else return fib(n-1) + fib(n-2);
9 }
10 int main() {
11     Number<fib(5)> n;
12 }
```



Affichage souhaité du Code 6

```
1 8
```

Affichage produit par le Code 6

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 7

```
1 #include <iostream>
2 #include <list>
3 template<class T> struct MyList {
4     std::list<T>& v;
5     MyList(std::list<T>& v) : v(v) {}
6 };
7 template<class Container, class Fct> void forall(Container c, Fct f) {
8     for(auto e : c)
9         f(e);
```



```
10 }
11 int main() {
12     std::list<int> l = {0, 1, 2, 3, 4, 5};
13     MyList<int> myl(l);
14     forall(myl, [](int i) { std::cout << i << " " });
15 }
```

Affichage souhaité du Code 7

```
1 0 1 2 3 4 5
```

Affichage produit par le Code 7

```
1 Erreur de compilation
```

Code 8

```
1 #include <iostream>
2
3 double operator "" ms(long double x) { return 0.001 * x; }
4 double operator "" s(long double x) { return x; }
5
6 int main() {
7     std::cout << "1ms is " << 1ms << " s" << std::endl;
8 }
```



Affichage souhaité du Code 8

```
1 1ms is 0.001 s
```

Affichage produit par le Code 8

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 9

```
1 #include <iostream>
2 #include <vector>
3 class MyVector
4 {
5     std::vector<int> v;
6 public:
7     MyVector(int i = 0) : v(std::vector<int>(i)) {}
8     void add(int i) { v.push_back(i); }
9     void print()
10    {
11        for(int i : v)
12            std::cout << i << " ";
13        std::cout << std::endl;
14    }
15};
16
17 int main()
18{
19    std::vector<MyVector> vector(5); int n = 4;
20    for(auto it = vector.begin(); it != vector.end(); it++)
21    {
22        MyVector v;
23        for(int i = n; i >= 0; i--)
24            v.add(i);
25        *it = v; n--;
26    }
27    std::sort(vector.begin(), vector.end()); //trie les vector par taille croissante
28    for(MyVector v : vector)
29        v.print();
30}
```



Affichage souhaité du Code 9

```
1 0
2 0 1
3 0 1 2
4 0 1 2 3
5 0 1 2 3 4
```

Affichage produit par le Code 9

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 10

```
1 #include <iostream>
2 using namespace std;
3
4 struct Mere
5 {
6     void f(int n) {cout << "Mere::integer" << n << endl; }
7     void f(char n) {cout << "Mere::character" << n << endl; }
8 };
9
10 struct Fille : public Mere
11 {
12     void f(int n, int m)
13     {
14         cout << "Fille::integers" << n << " " << m << endl;
15     }
16 };
17
18 int main()
19 {
20     int n = 1;
21     char c = 'a';
22     Fille b;
23     b.f(n);
24     b.f(c); 
25     b.f(n, c);
26 }
```

Affichage souhaité du Code 10

```
1 Mere::integer
2 Mere::character
3 Fille::integers
```

Affichage produit par le Code 10

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 11

```
1 #include <iostream>
2 using namespace std;
3
4 struct Poulet
5 {
6     Poulet() { cout << "+_"; }
7     Poulet(const A& a) { cout << "r_"; }
8     ~Poulet() { cout << "-_"; }
9     Poulet& operator =(const Poulet& a)
10    {
11        cout << "=_" << endl;
12        return *this;
13    }
14};
15
16 void f(Poulet a) {}
17 void g(Poulet& a) {}
18
19 int main()
20 {
21     Poulet a;
22     f(a);
23     Poulet * aa = new Poulet();
24     aa = &a;
25     g(*aa); 
26 }
```

Affichage souhaité du Code 11

```
1 + r - + = - -
```

Affichage produit par le Code 11

```
1 + r - + -
```

Matricule :

Nom :

Prénom :

Groupe :

Code 12

```
1 #include <iostream>
2 class A
3 {
4     protected:
5         int i;
6     public:
7         A(int i = 0) : i(i) {}
8         void print() { std::cout << i << std::endl; }
9     };
10 class B : public A
11 {
12     int j;
13     public:
14         B(int i = 0) : A(i + 2), j(i + 3) {}
15         void print() { std::cout << (i + j) << std::endl; }
16     };
17 int main() { A a = B(); a.print(); }
```



Affichage souhaité du Code 12

1 5

Affichage produit par le Code 12

1 2

Code 13

```
1 #include <iostream>
2 struct A { void print() { std::cout << "A" << std::endl; } }
3 struct B : A { void print() { std::cout << "B" << std::endl; } }
4 int main() {
5     B b; A & a = b;
6     a.print();
7 }
```



Affichage souhaité du Code 13

1 B

Affichage produit par le code 13

1 A

Matricule :

Nom :

Prénom :

Groupe :

Code 14

```
1 #include <iostream>
2
3 template<class T = int> class A
4 {
5     T t;
6
7     public:
8         A(const T& t = T()) : t(t) {}
9         T& operator() () { return t; }
10    };
11
12 int main()
13 {
14     A<int> i(2);
15     A<double> f(2.2);
16     std::cout << i() << " " << f() << std::endl;
17
18     i = f;
19     std::cout << i() << " " << f() << std::endl;
20 }
```



Affichage souhaité du Code 14

```
1 2 2.2
2 2
```

Affichage produit par le code 14

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 15

```
1 struct Duck { void quack() {} };
2
3 struct Pig { void groink() {} };
4
5 template<class T> auto test_duck(const T& t, int)
6     -> decltype(t.quack(), bool())
7 { return true; }
8
9 template<class T> auto test_duck(const T& t, long)
10 { return false; }
11
12 template<class T> bool is_duck(const T& t)
13 {
14     return test_duck(t, 0); 
15 }
16
17 int main()
18 {
19     Duck duck; Pig pig;
20
21     cout << is_duck(duck) << "\n";
22     cout << is_duck(pig) << endl;
23 }
```

Affichage souhaité du Code 15

```
1 0
```

Affichage produit par le code 15

```
1 0 0
```

Matricule :

Nom :

Prénom :

Groupe :

Code 16

```
1 #include <iostream>
2 #include <type_traits>
3
4 template<class T>
5 using isNotInt = std::enable_if_t<! std::is_same_v<T, int>>;
6
7 template<class T>
8 class A
9 {
10     T t;
11     int i;
12
13 public:
14     A(int i) : i(i) { cout << i << "\n"; }
15
16     template<class = isNotInt<T>>
17     A(T t) : t(t), i(-1) { cout << i << "\n"; }
18 };
19
20 int main()
21 {
22     A<double> a(1.1); //ok
23     A<int> b(1); //ok
24 }
```



Affichage souhaité du Code 16

```
1 1.1 1
```

Affichage produit par le code 16

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 17

```
1 #include <iostream>
2 #include <string>
3 #include <type_traits>
4
5 struct Duck
6 {
7     std::string name;
8     Duck(std::string name) : name(name) {}
9     void quack() const { cout << name << ":~quack~!"; }
10 };
11
12 template<class>
13 struct sfinae_true : std::true_type {};
14
15 template<class T>
16 static auto test_duck(int)
17     -> sfinae_true<decltype(T().quack())>; 
18
19 template<class>
20 static auto test_duck(long)
21     -> false_type;
22
23 template<class T>
24 struct bool_duck : decltype(test_duck<T>(0)) {};
25
26 int main()
27 {
28     cout << (bool_duck<Duck>() == true) << endl;
29 }
```

Affichage souhaité du Code 17

1

Affichage produit par le Code 17

0

Matricule :

Nom :

Prénom :

Groupe :

Code 18

```
1 #include <iostream>
2
3 struct A { void a() { std::cout << "A" << std::endl; } };
4
5 struct B {
6     int;
7     void b() { cout << "B" << endl; }
8 };
9
10 template<class T> bool isA(T t) { return sizeof(t) == sizeof(A); }
11
12 template<class T> bool isB(T t) { return sizeof(t) == sizeof(B); }
13
14 template<class T> void f(T t)
15 {
16     if(isA(t))
17         t.a();
18     else
19         t.b();
20 }
21
22 int main()
23 {
24     A a; B b;
25     f(a); f(b);
26 }
```



Affichage souhaité du Code 18

1 1

Affichage produit par le Code 18

1 0

Matricule :

Nom :

Prénom :

Groupe :

Code 19

```
1 #include <iostream>
2 using namespace std;
3 struct A {
4     A() { cout << "+A" << endl; }
5     A(const A& a) { cout << "rA" << endl; }
6     virtual ~A() { cout << "-A" << endl; }
7 };
8 struct B {
9     B() { cout << "+B" << endl; }
10    B(const B& b) { cout << "rB" << endl; }
11    virtual ~B() { cout << "-B" << endl; }
12 };
13 struct C : public A, public virtual B {
14     C() { cout << "+C" << endl; }
15     C(const C& c) { cout << "rC" << endl; }
16     virtual ~C() { cout << "-C" << endl; }
17 };
18
19 void f(A a) {}
20
21 int main() {
22     C c; cout << endl;
23     f(c); cout << endl;
24 }
```

Affichage souhaité du Code 19

```
1 +B    rA    -C    -C
2 +A        -A    -A
3 +C    -B    -B
```



Affichage produit par le Code 19

```
1 +A    rA    -C    -C
2 +B    rB    -B    -B
3 +C    rC    -A    -A
```

Remarque 1. Les affichages ci-dessus sont à lire en colonnes, de haut en bas et de gauche à droite. Les lignes vides sont à ignorer.

Matricule :

Nom :

Prénom :

Groupe :

Code 20

```
1 #include <iostream>
2 #include <vector>
3 struct A {
4     int i;
5     A(int i) : i(i) {}
6     void print() { std::cout << i << "\u033"; }
7 };
8
9 struct B
10 {
11     A a;
12     B(A a) { this->a = a; }
13     void print() { std::cout << a.i << "\u033"; }
14 }
15
16 int main() {
17     A a(1); B b(a);
18     b.print();
19 }
```



Affichage souhaité du Code 20

```
1 1
```

Affichage produit par le Code 20

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 1

```
1 #include <iostream>
2
3 class Integer
4 {
5     unsigned i;
6     bool positive;
7
8 public:
9     Integer(unsigned i, bool positive = true) : i(i), positive(positive) {}
10    Integer operator +(Integer a)
11    {
12        if(positive && a.positive)
13            return Integer(i + a.i);
14        else if(positive && !a.positive)
15            return Integer(i - a.i);
16        else if(!positive && a.positive)
17            return Integer(a.i - i);
18        else
19            return Integer(-i - a.i);
20    }
21
22    friend std::ostream& operator << (std::ostream& out, const Integer& i)
23    {
24        if(!i.positive)
25            out << "-";
26        out << i.i;
27    }
28};
29
30 int main()
31 {
32     Integer a1(2u);
33     Integer a2(3u, false);
34
35     Integer s = a1 + a2;
36     std::cout << s << std::endl;
37 }
```



Affichage souhaité du Code 1

1 -1

Affichage produit par le Code 1

1 4294967295

Matricule :

Nom :

Prénom :

Groupe :

Code 2

```
1 #include <iostream>
2 using namespace std;
3
4 class exceptA {};
5 class exceptB : public exceptA {};
6 class exceptC : public exceptB {};
7
8 void f() { throw exceptB(); }
9
10 int main()
11 {
12     try
13     {
14         f();
15     }
16     catch(exceptA& e)
17     {
18         cout << "I caught an A" << endl;
19     }
20     catch(exceptB& e)
21     {
22         cout << "I caught a B" << endl;
23     }
24     catch(exceptC& e)
25     {
26         cout << "I caught a C" << endl;
27     }
28     catch(...)
29     {
30         cout << "I caught something" << endl;
31     }
32 }
```



Affichage souhaité du Code 2

1 I caught a B

Affichage produit par le Code 2

1 I caught an A

Matricule :

Nom :

Prénom :

Groupe :

Code 3

```
1 #include <iostream>
2 #include <vector>
3 struct A {
4     int i;
5     A(int i) : i(i) {}
6     void print() { std::cout << i << "\u033"; }
7 };
8 int main() {
9     std::vector<A> v(5);
10    for(int j = 0; j < v.size(); j++)
11        v[j].i = j;
12    for(A a : v)
13        a.print();
14 }
```



Affichage souhaité du Code 3

```
1 0 1 2 3 4
```

Affichage produit par le Code 3

```
1 Erreur de compilation
```

Code 4

```
1 #include <iostream>
2 struct A {
3     int i;
4     A operator +(int i) { std::cout << "For_Maccrage_we_march" << std::endl; }
5     A operator +(A a) { std::cout << "and_we_shall_know_no_fear" << std::endl; }
6 };
7 int main() {
8     int i1; A a1; A a2;
9     i1 + a1 + a2; //vous ne pouvez pas changer cette ligne
10 }
```



Affichage souhaité du Code 4

```
1 For Maccrage we march
2 and we shall know no fear
```

Affichage produit par le Code 4

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 5

```
1 #include <iostream>
2 class A {};
3 struct
4 {
5     A * a;
6     B() : a(new A()) {}
7     ~B() { delete a; } 
8     void print() { cout << "Get_to_the_choppaaa" << endl; }
9 };
10 
11 void f(B b) { std::cout << "You have no respect for logic" << std::endl; }
12
13 int main()
14 {
15     B b;
16     f(b);
17     b.print();
18
19     B bb;
20     bb = b;
21     bb.print();
22
23     bb = bb;
24     bb.print();
25 }
```

Affichage souhaité du Code 5

```
1 You have no respect for logic
2 Get to the choppaaa
3 Get to the choppaaa
4 Get to the choppaaa
```

Affichage produit par le Code 5

```
1 Erreur de segmentation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 6

```
1 #include <iostream>
2 template<class T> struct Poney {
3     T & t; int & i;
4     Poney(T & t) { this->t = t; }
5     Poney(int & i) { this->i = i; }
6     void print() { std::cout << t << " " << i << std::endl; }
7 };
8 int main() {
9     Poney<int> b(2); b.t = 3; 
```


10 b.print();
11 }

Affichage souhaité du Code 6

1 3 2

Affichage produit par le Code 6

1 Erreur de compilation //il y a potentiellement deux erreurs dans ce code

Code 7

```
1 #include <iostream>
2 template<class T> struct MyVector {
3     std::vector<T> v;
4     MyVector(std::vector<T>& v) : v(v) {}
5 };
6 template<class Container, class Fct> void forall(Container c, Fct f) {
7     for(auto e : c)
8         f(e); 
```


9 }
10 int main() {
11 std::vector<int> v = {0, 1, 2, 3, 4, 5};
12 MyVector<int> myv(v);
13 forall(myv, [](int i) { std::cout << i << " "; });
14 }

Affichage souhaité du Code 7

1 0 1 2 3 4 5

Affichage produit par le Code 7

1 Erreur de compilation

Matricule :

Nom :

Prénom :

Groupe :

Code 8

```
1 #include <iostream>
2 #include <list>
3 #include <algorithm>
4
5 class MyList
6 {
7     std::list<int> l;
8 public:
9     MyList(int i = 0) : l(std::list<int>(i)) {}
10    void add(int i) { l.push_back(i); }
11    void print()
12    {
13        for(int i : l)
14            std::cout << i << " ";
15        std::cout << std::endl;
16    }
17};
18
19 int main()
20 {
21     std::list<MyList<int>> list(5);
22     for(auto it = list.begin(), int n = 4; it != list.end(); it++, n--)
23     {
24         MyList<int> l;
25         for(int i = n; i >= 0; i--)
26             l.add(i);
27         *it = l;
28     }
29     std::sort(list.begin(), list.end()); //trie les listes par taille croissante
30     for(MyList<int> l : list)
31         l.print();
32 }
```



Affichage souhaité du Code 8

```
1 0
2 0 1
3 0 1 2
4 0 1 2 3
5 0 1 2 3 4
```

Affichage produit par le Code 8

```
1 Erreur de compilation
```

Matricule :

Nom :

Prénom :

Groupe :

Code 9

```
1 #include <iostream>
2 template<int N> struct Number {
3     int n;
4     Number() : n(N) { std::cout << n << std::endl; }
5 };
6 int fib(int n) {
7     if(n == 0 || n == 1) return 1;
8     else return fib(n-1) + fib(n-2);
9 }
10 int main() {
11     Number<fib(5)> n;
12 }
```

Affichage souhaité du Code 9

1 8

Affichage produit par le Code 9

1 Erreur de compilation

Code 10

```
1 #include <iostream>
2 int f(const char tab[]) {
3     return sizeof(tab) / sizeof(*tab);
4 }
5 int main()
6 {
7     char bat[] = {1, 2, 3, 4, 5, 6, 7};
8     std::cout << (sizeof(bat) / sizeof(*bat)) << std::endl;
9     std::cout << f(bat) << std::endl;
10 }
```

Affichage souhaité du Code 10

1 7
2 7

Affichage produit par le Code 10

1 7
2 8

Matricule :

Nom :

Prénom :

Groupe :

Code 11

```
1 #include <iostream>
2 using namespace std;
3
4 struct Mere
5 {
6     void f(int n) {cout << "Mere::integer" << n << endl; }
7     void f(char n) {cout << "Mere::character" << n << endl; }
8 };
9
10 struct Fille : public Mere
11 {
12     void f(int n, int m)
13     {
14         cout << "Fille :: integers" << n << " " << m << endl;
15     }
16 };
17
18 int main()
19 {
20     int n = 1;
21     char c = 'a';
22     Fille b;
23     b.f(n);
24     b.f(c);
25     b.f(n, c);
26 }
```



Affichage souhaité du Code 11

```
1 Mere :: integer
2 Mere :: character
3 Fille :: integers
```

Affichage produit par le Code 11

```
1 Erreur de compilation
```

Code 12

```

1 #include <iostream>
2 using namespace std;
3
4 class Licorne
{
5     public:
6         Licorne() { cout << "+_"; }
7         Licorne(const A& a) { cout << "r_"; }
8         ~Licorne() { cout << "-_"; }
9         Licorne& operator =(const Licorne& a)
10        {
11            cout << "=_" << endl;
12            return *this;
13        }
14    };
15
16 void f(Licorne a) {}
17 void g(Licorne& a) {}
18
19 int main()
20 {
21     Licorne a;
22     f(a);
23     Licorne * aa = new Licorne();
24     aa = &a;
25     g(*aa); 
26 }
27

```

Affichage souhaité du Code 12

1 + r - + = - -

Affichage produit par le code 12

1 + r - + -

Matricule :

Nom :

Prénom :

Groupe :

Code 13

```
1 #include <iostream>
2 struct Balle {};
3 struct Joueur
4 {
5     Joueur* cible;
6     Joueur(Joueur* j = nullptr) : cible(j) {}
7     void lance(Balle ball)
8     {
9         try
10        {
11            throw ball;
12        }
13        catch(Balle& b)
14        {
15            std::cout << "J'ai attrapé la balle" << std::endl;
16            cible->lance(b);
17        }
18    }
19};
20
21 int main()
22 {
23     Joueur p1;
24     Joueur p2(&p1);
25     p1.cible = &p2;
26
27     Balle b;
28     p1.lance(b);
29 }
```



Affichage souhaité du Code 13

```
1 J'ai attrapé la balle
2 J'ai attrapé la balle
```

Affichage produit par le code 13

```
1 Erreur d'exécution
```

Question 2. Expliquez la différence entre la ligature statique et la ligature dynamique des liens, et la façon de mettre en œuvre ces différents concepts. /10

Question 3. Expliquez les différences principales entre le mécanisme des generics en Java et les templates en C++. /10

Code 1

```
1 #include <iostream>
2 using namespace std ;
3
4 int main()
5 {
6     for(unsigned i = 4; i >= 0; i--)
7         cout << i << " ";
8         cout << "BOOM" << endl ;
9 }
10
```

Car le i est de type unsigned et ne peut donc pas aller en négatif, ici on ira en négatif car ≥ 0 puis faire le $--$.

Affichage souhaité du Code 1

```
1 4 3 2 1 0 BOOM
```

Affichage produit par le code 1

```
1 Boucle infinie
```

Code 2

```
1 #include <iostream>
2
3 int f()
4 {
5     static int i { 4 };
6     return --i;
7 }
8
9 int main()
10 {
11     std::cout << f() << std::endl;
12     while (f()) {
13         std::cout << f() << std::endl;
14     }
15 }
```

Il faut retourner $--i$ désincrément d'abord, il faut écrire $i--$.

Affichage souhaité du Code 2

```
1 4
2 -1
```

Affichage produit par le code 2

```
1 3
2 -1
```

Code 3

```
1 #include <iostream>
2
3 struct A
4 {
5     int i;
6     A(int i) { this->i = i; }
7 };
8
9 struct B
{
10     A a;
11     B(A a) { this->a = a; }
12 };
13
14
15 int main()
16 {
17     A a(2); B b(a);
18     std::cout << b.a.i << std::endl;
19 }
```

Dans B on ne donne pas de valeur à A. Le seul constructeur pour A a besoin d'un int en paramètre. Ici on déclare un A sans int. Donc on doit écrire A a = A(int); avec un int à la place de int.

Affichage souhaité du Code 3

```
1 2
```

Affichage produit par le code 3

```
1 Erreur de compilation
```

Code 4

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a1 = 2.1;
7     cout << a1 << "\n";
8     int a2{2.1};
9     cout << a2 << "\n";
10    int a3{2.1};
11    cout << a3 << endl;
12 }
```

Entre accolade c'est pour indiquer qu'on ne veut pas de conversion mais un double ne peut pas entrer dans un int.

Affichage souhaité du Code 4

```
1 2 2 2
```

Affichage produit par le code 4

```
1 Erreur de compilation
```

Code 5

```
1 #include <iostream>           Car on échange les adresses, à la ligne 6 on dit que
2 void swap(int * i, int * j) l'adresse de i prend l'adresse de j et l'adresse de j
3 {                           prend l'adresse de temp.
4     int * tmp = i;
5     i = j; j = tmp;
6 }
7
8 int main()
9 {
10    int i = 2; int j = 3; swap(&i, &j);
11    std::cout << i << " " << j << std::endl;
12
13 }
```

Affichage souhaité du Code 5

```
1 3 2
```

Affichage produit par le code 5

```
1 2 3
```

Code 6

```
1 #include <iostream>
2 #include <list>
3
4 int main()
5 {
6     std::list<int> l;
7     for(int i = 0; i <= 6; i++)
8         l.push_back(i * i - i);
9     for(int i = 0; i < l.size(); i++)
10    {
11        std::cout << l.back() << " "; //affiche dernier elem
12        l.pop_back(); //supprime dernier elem
13    }
14 }
```

Ligne 9 dans le for, à chaque passage il test la taille de la liste mais celle-ci diminue car on supprime à chaque fois le dernier élément.

Affichage souhaité du Code 6

```
1 30 20 12 6 2 0 0
```

Affichage produit par le code 6

```
1 30 20 12 6
```

Code 7

FICHIER A.CPP

```
1 #include <iostream>
2
3 class B;
4
5 class A
{
6     public:
7         B * b;
8         A() : b(nullptr) {}
9         void print()
10        {
11            std::cout << "A_with_B" << std::endl;
12        }
13    };
14 }
```

FICHIER B.CPP

```
1 #include <iostream>
2
3 #include "A.cpp"
4
5 class B
{
6     public:
7         A a;
8         B() {}
9         void print()
10        {
11            std::cout << "B_with_A" << std::endl;
12        }
13    };
14 }
```

FICHIER ABMAIN.CPP

```
1 #include <iostream>
2 #include "A.cpp"
3 #include "B.cpp"
4
5 int main()
{
7     B b; A a;
8     a.b = &b; b.a = a;
9     a.print(); b.print();
10}
```

Supprimer le include "A.cpp" car il est déjà
include dans B, du coup sa boucle

Affichage souhaité du Code 7

```
1 A with B
2 B with A
```

Affichage produit par le code 7

```
1 Erreur de compilation
```

Code 8

```
1 #include <iostream>
2 struct A
3 {
4     int i;
5     A() : i(5) {}
6 };
7 struct B
8 {
9     A * a;
10    B() : a(new A()) {} //vous ne pouvez pas modifier cette ligne
11    ~B() { delete a; }
12 };
13
14 void f(B b) { std::cout << "I'm evil" << std::endl; }
15
16 int main()
17 {
18     B b; f(b);
19     cout << g.a-> i << endl;
20 }
```

Ligne 19 il manque le std:: ensuite à la ligne 18 on a pas besoin d'appeler la fonction f() car elle n'est pas dans l'affichage souhaité, pour finir, à la ligne 19, ce n'est pas g.a->i mais b.a->i g n'existe pas

Affichage souhaité du Code 8

```
1 5
```

Affichage produit par le code 8

```
1 Erreur de segmentation
```

Code 9

```
1 #include <iostream>
2 int main()
3 {
4     int i = 5; int j = 2; double d = 1.25; //vous ne pouvez pas modifier cette ligne
5     std::cout << (i / j + d) << std::endl;
6 }
```

avant de print j'ajoute
double div = ((double)i)/ j;
puis je print (div + d)

Affichage souhaité du Code 9

```
1 3.75
```

Affichage produit par le code 9

```
1 3.25
```

Code 10

```
1 #include <iostream>
2 int f(const char tab[])
3 {
4     return sizeof(tab) / sizeof(*tab);
5 }
6
7 int main()
8 {
9     char tab[] = {1, 2, 3, 4, 5, 6, 7};
10    std::cout << (sizeof(tab) / sizeof(*tab)) << " - ";
11    std::cout << f(tab) << std::endl;
12 }
```

Calcul la taille de l'adresse. Ici la machine est en 64bits donc 8 bytes.

Affichage souhaité du Code 10

```
1 7 7
```

Affichage produit par le code 10

```
1 7 8
```

Code 11

```
1 #include <iostream>
2
3 void plus_one(int& i) { std::cout << i + 1 << std::endl; }
4
5 int main()
6 {
7     plus_one(2);          plus_one attend une adresse et un immédiat n'as pas
8 }                                d'adresse. Donc dans le main on doit ajouter : int nb = 2; et
                                         puis appeler la fonction : plus_one(nb)
```

Affichage souhaité du Code 11

```
1 3
```

Affichage produit par le code 11

```
1 Erreur de compilation
```

Code 12

```
1 #include <iostream>          manque std:: ensuite en enlève le pointeur, donc on
2 static int* count;          enlève les* des lignes 3, 7, 12 et 19 car c'est static
3
4 struct Un_Brol
5 {
6     Un_Brol() { *count++; }
7 };
8
9 struct Pas_Un_Brol
10 {
11     Pas_Un_Brol() { *count++; }
12 };
13
14 int main()
15 {
16     Un_Brol a; Un_Brol b; Un_Brol c;
17     Pas_Un_Brol d; Pas_Un_Brol e;
18     cout << *count << endl;
19 }
20
```

Affichage souhaité du Code 12

```
1 5
```

Affichage produit par le code 12

```
1 Erreur de segmentation
```

Code 13

```
1 #include <iostream>
2
3 int min(const std::vector<int>& tab)
4 {
5     int min;
6     for(int i = 0; i < tab.size(); i++)
7         if(min > tab[i])
8             min = tab[i];
9
10    return min;
11 }
12
13 int main()
14 {
15     vector<int> v = {4, 3, 1, 8, 5, 9, 10};
16 }
```

Dans le main on appelle pas min. Ensuite, ligne 5 on ne donne pas de valeur à min donc prendra une valeur random dans la pile.

Affichage souhaité du Code 13

```
1 1
```

Affichage produit par le code 13

```
1 -150
```

Code 14

```
1 #include <iostream>
2 #include <string>
3
4 const char* hello()
5 {
6     const char str[] = "Hello\u00a9 there\u00a9!";
7     return str;
8 }
9
10 int main()
11 {
12     std::string s(hello());
13     std::cout << s << std::endl;
14 }
```

disparaît après le bloc, donc on doit l'écrire en dessous des include.

Affichage souhaité du Code 14

```
1 Hello there !
```

Affichage produit par le code 14

```
1 Erreur de segmentation
```

Code 15

FICHIER A.CPP

```
1 #ifndef A_H
2 #define A_H
3 #include <iostream>
4 #include "B.cpp"
5
6 struct A
7 {
8     B b;
9     A() {}
10    void print() { std::cout << "A with B" << std::endl; }
11};
12#endif
```

On ajoute struct B;
Du coup le B sera un pointeur donc ligne 8 -> B*b;

FICHIER B.CPP

```
1 #ifndef B_H
2 #define B_H
3 #include <iostream>
4 #include "A.cpp"
5
6 struct B
7 {
8     A a;
9     B() {}
10    void print() { std::cout << "B with A" << std::endl; }
11};
12#endif
```

FICHIER ABMAIN.CPP

```
1 #include <iostream>
2 #include "A.cpp"
3 #include "B.cpp"
4
5 int main()
6 {
7     B b; A a;
8     a.b = b; b.a = a;
9     a.print(); b.print();
10}
```

Supprimer la ligne 2, le #include "A.cpp" car B inclut déjà le A.
Du coup comme b est un pointeur, à la ligne 8 on doit mettre & --> a.b = &b

Affichage souhaité du Code 15

```
1 A with B
2 B with A
```

Affichage produit par le code 15

```
1 Erreur de compilation
```

Nom :

Prénom :

Groupe :

Matricule :

Code 1

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 class A
6 {
7     public:
8         ~A() { cout << "Boom" << endl; }
9 };
10
11 class B
12 {
13     vector<A*> v;
14     public :
15         B() : v(vector<A*>(4)) {}           for(unsigned i(0);i < v.size();i++){
16     };                                         A a;
17                                         v.at(i) = &a;
18 void f() { B b; }                                     }
19
20 int main()
21 {
22     f();
23     cout << "I_want_you_in_my_room" << endl << "(outdated_song)" << endl;
24 }
```

Il faut créer 4 objets A et donner leurs références aux pointeurs se trouvant dans le vecteur

A la fin de l'instruction, un destructeur sera appelé afin de libérer la mémoire, c'est pour cela que Boom sera affiché 4 fois

Affichage souhaité du Code 1

```
1 Boom
2 Boom
3 Boom
4 Boom
5 I want you in my room
6 (outdated song)
```

Code 2

```
1 #include <iostream>
2
3 int main()
4 {
5     int i = 33;
6     int const & ri = i;
7     std::cout << ri << std::endl;
8     i++;
9     std::cout << ri << std::endl;
10 }
```

La référence est constante, ce qui fait qu'on change uniquement la valeur qui s'y trouve et non pas l'adresse

Affichage souhaité du Code 2

```
1 33
2 34
```

Nom :

Prénom :

Groupe :

Matricule :

Code 3

```
1 #include <iostream>
2 using namespace std;
3
4 class A
5 {
6     public:
7         void f(int n) {cout << "A::integer" << n << endl; }
8         void f(char n) {cout << "A::character" << n << endl; }
9 };
10
11 class B : public A
12 {
13     public:
14         void f(int n, int m) { cout << "B::integers" << n << " " << m << endl; }
15 };
16
17
18 int main()
19 {
20     int n = 1;
21     char c = 'a';
22     B b;
23     b.f(n);           b.A:f(n);      Il faut qualifier à quelle classe appartienne ces mêmes méthodes
24     b.f(c);           b.A:f(c);      Préciser qu'on souhaite accéder à la méthode de la classe mère
25     b.f(n, c);
26 }
```

Affichage souhaité du Code 3

```
1 A::integer
2 A::character
3 B::integers
```

Code 4

```
1 #include <iostream>
2 int f() {
3     static int i { 4 };    Il faut changer la valeur par 5
4     return --i;           ou changer
5 }
6 int main() {
7     std::cout << f() << std::endl;
8     while (f()) { }
9     std::cout << f() << std::endl;
10}
```

Affichage souhaité du Code 4

```
1 4
2 -1
```

Nom :

Prénom :

Groupe :

Matricule :

Code 5

```
1 #include <iostream>
2
3 using namespace std;
4
5 class Integer
6 {
7     unsigned i;
8     bool positive;
9
10    public:
11        Integer(unsigned i, bool positive = true) : i(i), positive(positive) {}
12        Integer operator +(Integer a)
13        {
14            if(positive && a.positive) return Integer(i + a.i);
15            else if(positive && !a.positive) return Integer(i - a.i);
16            else if(!positive && a.positive) return Integer(a.i - i);
17            else return Integer(-i - a.i);
18        }
19
20    void print()
21    {
22        if(!positive)
23            cout << "-";
24        cout << i << endl;
25    }
26};
27
28 int main()                                Retirer les unsigned pour qu'on puisse accéder aux valeurs négatives
29 {                                         Avec les unsigned, une fois qu'on décremente en dessous de 0, on
30     Integer a1(2u);                      accède à la valeur maximal d'un unsigned
31     Integer a2(3u, false);
32
33     Integer s = a1 + a2;
34     s.print();
35 }
```

Affichage souhaité du Code 5

```
1 -1
```

Nom :

Prénom :

Groupe :

Matricule :

Code 6

```
1 #include <iostream>
2 using namespace std;
3
4 class exceptA {};
5 class exceptB : public exceptA {};
6 class exceptC : public exceptB {};
7
8 void f() { throw exceptB(); }
9
10 int main()
11 {
12     try
13     {
14         f();
15     }
16     catch(exceptA& e)
17     {
18         cout << "I caught an A" << endl;
19     }
20     catch(exceptB& e)
21     {
22         cout << "I caught a B" << endl;
23     }
24     catch(exceptC& e)
25     {
26         cout << "I caught a C" << endl;
27     }
28     catch(...)
29     {
30         cout << "I caught something" << endl;
31     }
32 }
```

Comme l'exception de la classe mère est prioritaire,
le code va directement accéder à son exception

Il faudra changer l'ordre des catchs en allant de la
classe fille jusqu'à la classe mère et non pas le
contraire

Affichage souhaité du Code 6

```
1 I caught a B
```

Nom :

Prénom :

Groupe :

Matricule :

Code 7

```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 class A
7 {
8     public:
9         int i; A(){} ; ou A(): i(0){};
10        A(int i) : i(i) {}
11        void print() { cout << i << endl; }
12    };
13
14 int main()
15 {
16     vector<A> v(5);
17     for(int j = 0; j < v.size(); j++)
18         v[j].i = j;
19     for(A a : v)
20         a.print();
21 }
```

Créer un constructeur de A qui recevra un entier pour instancier l'attribut i et donc remplir le vecteur par des nouveaux objets

Le constructeur par défaut qui ne reçoit pas d'argument et qui instancie la valeur de i ne l'instancie pas du tout

Ajouter un opérateur de comparaison qui va gérer les comparaisons entre objets A puisque que la méthode sort fait appelle à l'opérateur de comparaison afin de trier

On du vecteur, on fait appel au constructeur de la classe A pour chaque élément d'objet

Affichage souhaité du Code 7

```
1 0 1 2 3 4
```

Code 8

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5 class A {
6     public: A(){}
7     int i; A(int i){ this->i=i;}
8     bool operator <(A &a){ return this->i<a.i; }
9
10 int main()
11 {
12     vector<A> v = {5,4,3,2,1};
13     sort(v.begin(), v.end());
14     for(A a : v)
15         cout << a.i << " ";
16 }
```

Créer un constructeur de A qui recevra un entier pour instancier l'attribut i et donc remplir le vecteur par des nouveaux objets

Ajouter un opérateur de comparaison qui va gérer les comparaisons entre objets A puisque que la méthode sort fait appelle à l'opérateur de comparaison afin de trier

Affichage souhaité du Code 8

```
1 1 2 3 4 5
```

Code 2

FICHIER A.CPP

```

1 #include <iostream>
2
3 class B;
4
5 class A
{
6     public:
7         B * b;
8         A() : b(nullptr) {}
9         void print()
10        {
11            std::cout << "A_with_B" << std::endl;
12        }
13}
14

```

FICHIER B.CPP

```

1 #include <iostream>
2
3 #include "A.cpp"
4
5 class B
{
6     public:
7         A a;
8         B() {}
9         void print()
10        {
11            std::cout << "B_with_A" << std::endl;
12        }
13}
14

```

FICHIER ABMAIN.CPP

```

1 #include <iostream>
2 #include "B.cpp"
3 using namespace std;
4
5 int main()
{
    B b;
    A a;
    a.b = &b;
    b.a = a;
    a.print();
    b.print();
}

```

Il manque un include dans le main, sans include on ne peut pas accéder aux autres classes

Affichage souhaité du Code 2

```

1 A with B
2 B with A

```

Nom :

Prénom :

Groupe :

Matricule :

Code 3

```
1 #include <iostream>
2 int f(const char tab[])
3 {
4     return sizeof(tab) / sizeof(*tab);           Suite à l'appel de la fonction, le tableau passé en paramètre se voit
5 }                                              implicitement converti en pointeur de char
6                                         8/1 comme un char vaut 1 byte
7 int main()
8 {
9     char bat[] = {1, 2, 3, 4, 5, 6, 7};
10    std::cout << (sizeof(bat) / sizeof(*bat)) << std::endl;
11    std::cout << f(bat) << std::endl;
12 }
```

Le pointeur pointera vers le premier élément du tableau, tab vaut donc 8 puisque c'est sa taille en bytes

Affichage souhaité du Code 3

```
1 7
2 7
```

Remarque 1. Le code ci-dessus est exécuté sur un processeur 64 bits.

Code 4

```
1 #include <iostream>
2 #using namespace std;
3
4 int main()
5 {
6     int i = 5; int j = 2; double d = 1.5; //vous ne pouvez pas modifier cette ligne
7     cout << (i / j + d) << endl;
8 }                                              (double)i          un conversion explicite doit se faire à la variable i pour qu'on puisse faire une division
                                                               décimale
```

Affichage souhaité du Code 4

```
1 4
```

Nom :

Prénom :

Groupe :

Matricule :

Code 5

```
1 #include <iostream>
2 using namespace std;
3 class A {
4     public:
5         A() { cout << "+A"; }
6         A(const A& a) { cout << "rA"; }
7         ~A() { cout << "-A"; }
8         A& operator =(const A& a) { cout << "=A" << endl; return *this; }
9     };
10
11 void f(A a) {}
12 void g(A& a) {}  
On libère la mémoire allouée dynamiquement via la delete juste
13
14 int main()
15 {
16     A a;
17     f(a);
18     A * aa = new A(); delete aa;
19     aa = &a;
20     g(*aa);
21 }
```

Affichage souhaité du Code 5

1 +A rA -A +A -A -A

Code 6

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int a1 = 2.1;
7     cout << a1 << " ";
8     int a2(2.1);
9     cout << a2 << " ";
10    int a3{2.1};  
Conversion explicite nécessaire
11    cout << a3 << endl;
12 }
```

Affichage souhaité du Code 6

2 2 2

Nom :

Prénom :

Groupe :

Matricule :

Code 7

FICHIER A.CPP

```
1 #ifndef A_H
2 #define A_H
3 #include <iostream>
4 #include "B.cpp"
5
6 class A
7 {
8     public:
9         B b;
10        A() {}
11        void print()
12        {
13            std::cout << "A with B" << std::endl;
14        }
15    };
16#endif
```

FICHIER B.CPP

```
1 #ifndef B_H
2 #define B_H
3 #include <iostream>
4 #include "A.cpp"
5
6 class B
7 {
8     public:
9         A a;
10        B() {}
11        void print()
12        {
13            std::cout << "B with A" << std::endl;
14        }
15    };
16#endif
```

Ajouter un pointeur pour l'une de deux classes puisque ce sont des types incomplets

Nous n'avons pas connaissance de la taille d'un de ces objets alors que pour un pointeur c'est le cas

FICHIER ABMAIN.CPP

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     B b;
7     A a;
8     a.b = b;
9     b.a = a;
10    a.print();
11    b.print();
12 }
```

Include manquant et aucun problème d'inclusion multiple puisqu'il y a ifndef

Affichage souhaité du Code 7

```
1 A with B
2 B with A
```

Nom :

Prénom :

Groupe :

Matricule :

Code 1

```
1 #include <iostream>
2 int f(const char tab[]) {return sizeof(tab) / sizeof(*tab); }
3 int main()
4 {
5     char bat[] = {1, 2, 3, 4, 5, 6, 7};
6     std::cout << (sizeof(bat) / sizeof(*bat)) << std::endl;
7     std::cout << f(bat) << std::endl;
8 }
```

Code 2

```
1 #include <iostream>
2 using namespace std;
3
4 class A
5 {
6     public:
7     void f(int n) {cout << "A:: entier " << n << endl; }
8     void f(char n) {cout << "A:: char " << n << endl; }
9 };
10
11 class B : public A
12 {
13     public:
14     void f(int n, int m) { cout << "B:: entiers " << n << " " << m << endl; }
15 };
16
17
18 int main()
19 {
20     int n = 1;
21     char c = 'a';
22     B b;
23     b.f(n);
24     b.f(c);
25 }
```

Code 3

```
1 #include <iostream>
2 #include <utility>
3
4 int main()
5 {
6     std::pair<char, double> pcd( 'Z', -3.14);
7     std::pair<float, long> pfl = pcd;
8     std::cout << pcd.first << " " << pfl.second << std::endl;
9 }
```

Conversion implicite du code qui permet "Z" et "-3" après conversion en long int

Code 4

```
1 #include <iostream>
2 int f(const int tab[], int u) { return *(tab + u); }
3 int main()
4 {
5     int bat[] = {1,2,3,4,5,6,7};           Incrément de 1, la référence du pointeur de bat
6     std::cout << *(bat + 1) << std::endl;
7     std::cout << f(bat + 4, -2) << std::endl;   Incrément de 4 et ça décrément de 2 sur la
8 }                                         référence
                                              Affichage : 2 et 3
```

Nom :

Prénom :

Groupe :

Matricule :

Code 5

```
1 #include <iostream>
2
3 class Ball {};
4 class P
5 {
6     public:
7         P* target;
8         P(P* target) : target(target) {}
9         void aim(Ball b)
10    {
11        try
12        {
13            throw b;
14        }
15        catch(Ball& b)
16        {
17            std::cout << "Got it!" << std::endl;
18            target->aim(b);
19        }
20    }
21
22
23 int main()
24 {
25     P * parent = new P(0);
26     P * child = new P(parent);
27     parent->target = child;
28     parent->aim(Ball());
29 }
```

Boucle infini "Got it" suite à une fonction récursive

Code 6

```
1 #include <iostream>
2 int& f(int& i)
3 {
4     i *= 2;
5     return i;
6 }
7
8 int main()
9 {
10    auto i = 23;
11    std::cout << f(i)++ << std::endl; 46 à l'affichage
12    f(i) = 321.123; 47 après incrémentation puis passage à la méthode f qui va multiplier par 2 , la valeur de retour sera
13    std::cout << i << std::endl; écrasée par 321 puisque i est un entier
14 } on affiche ensuite 321
```

Code 7

```
1 #include <iostream>
2 #include <set>
3
4 int main()
5 {
6     std::multiset<int> msi = {3, 5, 3, 6, 2};  Affichage :
7     for(auto e : msi)
8         std::cout << e << " ";
9     std::cout << std::endl;
10 }
```

C'est un container qui permet de trier les valeurs qu'il stocke

Nom :

Prénom :

Groupe :

Matricule :

Code 8

```
1 #include <iostream>
2 using namespace std;
3
4 class exceptA {};
5 class exceptB : public exceptA {};
6 class exceptC : public exceptB {};
7
8 void f() { throw exceptB(); }
9
10 int main()
11 {
12     try
13     {
14         f();
15     }
16     catch(exceptA& e)
17     {
18         cout << "I caught an A" << endl;
19     }
20     catch(exceptB& e)
21     {
22         cout << "I caught a B" << endl;
23     }
24     catch(exceptC& e)
25     {
26         cout << "I caught a C" << endl;
27     }
28 }
```

Code 9

```
1 #include <iostream>
2 using namespace std;
3
4 class point{
5     protected:
6         int x, y;
7     public:
8         point(int a = 0, int b = 0) : x(a), y(b) {}
9         virtual void affiche()
10        { cout << "Je suis en " << x << ", " << y << endl; }
11    };
12
13 class pointcol : public point {
14     short col;
15     public:
16         pointcol(int a = 0, int b = 0, int c = 0) : point(a,b), col(c) {}
17         void affiche()
18         { cout << "Je suis en " << x << ", " << y << " et ma couleur "
19             << " est " << col << endl; }
20    };
21
22 int main()
23 {
24     point * p = new point(3,5);    pointcol * pc = new pointcol(8,6,2);
25     p = pc;                      je suis en 8,6 et ma couleur est 2
26     p->affiche();   pc->affiche();   je suis en 8,6 et ma couleur est 2
27     p = new point(3,5);    pc = (pointcol*)p;
28     p->affiche();   pc->affiche();
29 }                                je suis en 3,5
                                         je suis en 3,5
```

Nom :

Prénom :

Groupe :

Matricule :

Code 10

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main()
6 {
7     int t[] = {1,2,3,4};
8     vector<int> v(t, t);
9     cout << v.size() << " " << v.capacity() << endl;
10    for(int i = 0; i <= 8; i++)
11    {
12        v.push_back(0);
13        cout << v.size() << " " << v.capacity() << endl;
14    }
15 }
```

Code 11

```
1 #include <iostream>
2 using namespace std;
3
4 class A
5 {
6     public:
7         A() {cout << "+A" << endl; }
8         A(const A&) {cout << "cA" << endl; }
9         virtual ~A() {cout << "-A" << endl; }
10    };
11 class B : public A
12 {
13     public:
14         B() {cout << "+B" << endl; }
15         B(const B&) {cout << "cB" << endl; }
16         virtual ~B() {cout << "-B" << endl; }
17    };
18
19 void f(A)                                Affichage:
20 {                                         +A
21     cout << "f" << endl;                  +B
22 }                                         cA
23
24 int main()                                 f
25 {                                         -A
26     B b; f(b);                           -B
27 }
```

Code 12

```
1 #include <iostream>
2 using namespace std;
3 class A
4 {
5     public:
6         const int i;
7         A(int entier) {entier = i;}
8         A(int entier) : i(entier){};
9
10    int main()
11    {
12        A a(2.5); cout << a.i << endl;
13    }
```

Il faut initialiser les membres constants dans une liste d'initialisation

Code 13

```

1 #include <iostream>
2 using namespace std;
3
4 class A
{
5   public:
6     virtual A& operator=(const A&) { cout << "Aff_A" << endl; }
7 };
8 class B : public A
{
9   public:
10    virtual B& operator=(const B&) { cout << "Aff_B" << endl; }
11 };
12
13 int main()
14 {
15   A * a1 = new A(); A * a2 = new A();
16   B * b1 = new B(); B * b2 = new B();
17   b1 = b2; a1 = b1; a1 = b2;
18   a1 = new A(); a2 = new A();
19   b1 = new B(); b2 = new B();
20   *b1 = *b2; *a1 = *b1; *a1 = *b2;
21
22 }
23

```

Affichage:

AffB
AffA
AffA**Code 14**

```

1 #include <iostream>
2 using namespace std;
3
4 class A
{
5   public:
6     int a;
7     A() { cout << "Def_A" << endl; }
8     A(int a) : a(a) { cout << "+A" << endl; }
9 };
10
11 class B : public A
12 {
13   public:
14     int b;
15     B(int a = 2, int b = 1) : A(a), b(b) { cout << "+B" << endl; }
16
17 }
18
19 class C : public A
20 {
21   public:
22     int c;
23     C(int a = 3, int c = 2) : A(a), c(c) { cout << "+C" << endl; }
24
25
26 class D : public B, public C
27 {
28   public:
29     int d;
30     D(int a = 4, int b = 5, int c = 6, int d = 7) : B(b), C(c), d(d)
31     { cout << "+D" << endl; }
32
33
34 int main()
35 {
36   D d(1, 2, 3, 4);
37

```

Affichage:
+A
+B
+A
+C
+D