Nom : Prénom : N^o Étudiant :

Éléments de syntaxe C++

Structures de contrôle :

```
#include <iostream>
using namespace std;
int main () {
    ...
}
```

```
cin >> n;
cout << 3*n+1 << endl;
```

```
if ( x == 1 ) {
    ...;
} else {
    ...;
}
```

```
for ( int i=0; i < 10; i++ ) {
    ...;
}
```

```
while ( i <= 10 ) {
    ...;
}
```

```
do {
    ...;
} while ( i <= 10 );
```

```
int j;
...
switch (j) {
  case 1: res = "lundi"; break;
  case 2: res = "mardi"; break;
  ...
  default: res = "jour invalide";
}
```

Types de données :

```
struct AssureSocial {
  string nom, prenom, Nsecu;
  Date date_naissance;

  bool anniv(Date today) const; // méthode
};

AssureSocial litAssureSocial();
bool estMajeur(AssureSocial a);
bool AssureSocial::anniv(Date today) const {
  ... // définition de la méthode
}

AssureSocial a = litAssureSocial();
cout << a.nom << " " << a.prenom;
if (estMajeur(a)) ...
```

```
enum class Jour {lundi, mardi,
  mercredi, ..., dimanche};
Jour d = Jour::mercredi;
int i = int(d);  // conversion Jour -> int
d = Jour(4);     // conversion int -> Jour
```

```
vector<int> v;
v[5] = 3;
v.push_back(7);     v.pop_back();
cout << v.back();

array<int, 10> a;
a[9] = 2;
cout << a.size();
```

Fonctions, opérateurs et tests :

```
/** La fonction blabla
 * @param n une donnee
 * @return un entier
 **/
int blabla(int n) {
    int res;
    ...
    return res;
}

TEST_CASE("Fonction blabla") {
  CHECK( blabla(...) == 1 );
  CHECK_FALSE( estMajeur(a) );
}
```

```
struct Rat { int num, den; };

ostream& operator<<(ostream& out, Rat a) {
  out << a.num << "/" << a.den;
  return out;
}

bool operator==(Rat a, Rat b) {
  return a.num == b.num and a.den == b.den;
}

bool operator<=(Rat a, Rat b) {
  return (a-b).num <= 0;
}
```