Université Paris I – Panthéon-Sorbonne
90, rue de Tolbiac – 75634 Paris Cedex 13

Masters MAEF, QEM, DU MMEF — March 2016
Midterm exam

Introduction to Modern C++

(English version)

No documents, calculators, or computers allowed
Duration: 1h30.

Note: Please write clear and concise answers. Make sure that punctuation, if any, is visible. Most questions can be treated independently.

Question 1. What are some advantages of C++?

Question 2. What is "compilation" and why is it needed?

Question 3. Explain the following command line:

```
g++ -o something another.cpp --std=c++11
```

Question 4. Explain type safety in a few lines.

Question 5. Describe the types associated with each variable:

1. `int A;`
2. `int& B = A;`
3. `int* C;`
4. `std::string D;`
5. `double E, F, G;`
6. `auto H = 3.14;`
7. `std::vector<double> J;`
8. `std::array<std::array<int,2>,2> K;`

Question 6. Consider the following function:

```
double f(double x) {
  return x*x*x;
}
```

What is the *type* and *value* of `f(6)`? Is `f` a pure function? We have `f(1625)=-3951671`: Explain!

Question 7. What should be the type of a `std::vector` that contains the values 3.14, 2.718, -1?

Question 8. Are these statements true or false, and why?

1. We always have `x == x`.
2. We always have `x + y > x` if `y > 0`.
3. If `x > 0` and `y > 0`, then `x+y > 0`.
4. If `x > 0` and `y > 0`, then `x*y > 0`.
5. A `for` loop always terminates.
6. We always have `x * (y + z) = x*y + x*z`.

Question 9. Write a `for` loop that prints the numbers from 0 to 9 included.

Question 10 . What does this program display?

```
for (int i = 4; i < 72; i = i * i) {
  std::cout << i << ", ";
}
int k = 42;
while (k > 0) {
  std::cout << k << ", ";
  k = k / 2 - 1;
}
```

Question 11. What does `#include <iostream>` do? Give two more examples of a compiler directive.

Question 12. Consider the following program:

```
#include <iostream>

int myFunction(int N) {
  if (N == 0) {
    return 0;
  }
  return N + myFunction(N - 1);
}
```

How do you call this situation, when a function calls itself? Name one advantage and one disadvantage of doing so. What does `myFunction(n)` compute?

Question 12. What does the following program print? Explain why in details. What is `&y`? What is `*z`?

```
int  x = 42;
int& y = x;
int* z = &y;
int  t = x;
y = 73;

std::cout << x << std::endl;
std::cout << y << std::endl;
std::cout << *z << std::endl;
std::cout << t << std::endl;
```

Question 13. Can it be always determined automatically whether a given program terminates?

Question 14. Recall the definition of a *field* and of a *method* of a class. By default, what is the visibility (private, public, ...) of class members?

Question 15. Write a class `Duck` that has fields `name`, `age`, `position` and methods `fly()` and `eat()`. Which of these should be private or public, and why?

Question 16. The class `Duck` inherits from `Bird` which itself inherits from `Animal`. What do these classes look like? Explain the interest of inheritance.

Question 17; What is a header file and why is it used for?

Question 18. Identify as many mistakes as you can in the following program (you can use line numbers as references) and give a short explanation.

```
#include <iostream >

struct Point2D {
  double x, y;
  double norm2()
};

double Point2D::norm2() {
  return x+y*y;
}

void main() {
  double x, y, z;
  Point2D v {10, 2};
```

```
  std::cout << v.norm2() << endl;
  std::cout << "Enter a number: ";
  std::cin << x;
  std ::cout << myFunction(x) << endl;
}

int myFunction(double x) {
  if (x = 0) {
    std::cout << "Don't divide by zero !" std::endl;
  }
  else {
    return 42/x;
  }
};
```

Question 19. In 2014, a vulnerability in Apple products was identified (CVE-2014-1266) which enabled an attacker to capture or modify data in sessions that should have been protected by SSL/TSL (i.e. secured by an HTTPS connection over the Internet).

Here is an extract of the code. There is a succession of security checks, which send the program to a `fail` if something goes wrong. Can you spot the mistake in this code? How to avoid making such mistakes?

```
. . .
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
. . .
```

Note: You can test your phone or computer for this vulnerability by visiting `https://gotofail.com/`.

Question 20. Display the 6-th element of vector `V`. Then sort `V`. Display the 6-th element of `V` again.

```
int main() {
  std::vector<int> V {3, 1, 4, 1, 5, 9, 2, 6};

  // Your code here
}
```

Université Paris I – Panthéon-Sorbonne
90, rue de Tolbiac – 75634 Paris Cedex 13

Masters MAEF, QEM, DU MMEF — Mars 2016
Examen de mi-parcours

Introduction au C++ moderne

(Version française)

Documents, calculateurs et ordinateurs interdits
Durée: 1h30.

Remarque: Merci de fournir des réponses claires et concises. Assurez-vous que la ponctuation, s'il y a lieu, est visible. La plupart des questions peuvent être traitées indépendamment des autres.

Question 1. Quels sont quelques avantages de C++ ?

Question 2. Qu'est ce que la "compilation" est pourquoi est-elle nécessaire ?

Question 3. Expliquez la commande suivante :

```
g++ -o something another.cpp --std=c++11
```

Question 4. Expliquez ce qu'est la sûreté de type (*type safety*) en quelques lignes.

Question 5. Décrivez les types associés à chacune de ces variables :

1. `int A;`
2. `int& B = A;`
3. `int* C;`
4. `std::string D;`
5. `double E, F, G;`
6. `auto H = 3.14;`
7. `std::vector<double> J;`
8. `std::array<std::array<int,2>,2> K;`

Question 6. Considérez la fonction qui suit :

```
double f(double x) {
  return x*x*x;
}
```

Quel est le *type* et la *valeur* de `f(6)`? Est-ce que `f` est une fonction pure ? Nous obtenons `f(1625)=-3951671`: Expliquez !

Question 7. Quel devrait être le type d'un `std::vector` contenant les valeurs 3.14, 2.718, -1 ?

Question 8. Parmi les affirmations suivantes, lesquelles sont vraies ou fausses, et pourquoi ?

1. On a toujours `x == x`.
2. On a toujours `x + y > x` si `y > 0`.
3. Si `x > 0` et `y > 0`, alors `x+y > 0`.
4. Si `x > 0` et `y > 0`, alors `x*y > 0`.
5. Une boucle `for` se termine toujours.
6. On a toujours `x * (y + z) = x*y + x*z`.

Question 9. Écrivez une boucle `for` qui affiche les nombres de 0 à 9 inclus.

Question 10 . Qu'affiche le programme suivant ?

```
for (int i = 4; i < 72; i = i * i) {
  std::cout << i << ", ";
}
int k = 42;
while (k > 0) {
  std::cout << k << ", ";
  k = k / 2 - 1;
}
```

Question 11. Que signifie `#include <iostream>` ? Donnez deux autres exemples d'une directive de compilation.

Question 12. Consider the following program:

```
#include <iostream>

int myFunction(int N) {
  if (N == 0) {
    return 0;
  }
  return N + myFunction(N - 1);
}
```

Comment s'appelle une telle fonction, qui s'appelle elle-même ? Mentionnez un avantage et un désavantage de cette approche. Que calcule `myFunction(n)` ?

Question 12. Qu'affiche le programme suivant ? Expliquez pourquoi en détails. Qu'est ce que `&y`? Qu'est ce que `*z`?

```
int  x = 42;
int& y = x;
int* z = &y;
int  t = x;
y = 73;

std::cout << x << std::endl;
std::cout << y << std::endl;
std::cout << *z << std::endl;
std::cout << t << std::endl;
```

Question 13. Peut-on déterminer de manière automatique si un programme termine, quel que soit ce programme ?

Question 14. Rappelez la définition d'un *champ* (*field*) et d'une *méthode* (*method*) de classe. Quelle est la visibilité (private, public, ...) par défaut des membres d'une classe ?

Question 15. Écrivez une classe `Canard` avec pour champs `name`, `age`, `position` et pour méthodes `fly()` et `eat()`. Lesquels doivent être déclarés private ou public, et pourquoi ?

Question 16. La classe `Canard` hérite de `Oiseau`, laquelle hérite à son tour de `Animal`. Décrivez ces classes. Quel est l'intérêt de l'héritage ?

Question 17; Qu'est-ce qu'un fichier d'en-tête (*header file*) et à quoi sert un tel fichier ?

Question 18. Repérez un maximum d'erreurs dans le programme suivant (vous pouvez utiliser les numéros de ligne pour y faire référence) en donnant une courte explication de chaque.

```
#include <iostream >

struct Point2D {
  double x, y;
  double norm2()
};

double Point2D::norm2() {
  return x+y*y;
}

void main() {
  double x, y, z;
```

```
  Point2D v {10, 2};
  std::cout << v.norm2() << endl;
  std::cout << "Enter a number: ";
  std::cin << x;
  std ::cout << myFunction(x) << endl;
}

int myFunction(double x) {
  if (x = 0) {
    std::cout << "Don't divide by zero !" std::endl;
  }
  else {
    return 42/x;
  }
};
```

Question 19. En 2014, une vulnérabilité dans les produits Apple a été découverte (CVE-2014-1266), qui permet à des attaquants d'intercepter et modifier les données de connexions normalement protégées par SSL/TSL (i.e. lors d'une connexion HTTPS sur Internet).

Voici un extrait du programme. Il y a une succession de vérifications, qui envoient le programme dans une section `fail` en cas de problème. Pouvez-vous identifier l'erreur dans ce programme ? Comment éviter une telle erreur?

```
. . .
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(...);
. . .
```

Remarque : Vous pouvez vérifier si votre téléphone ou ordinateur est vulnérable en visitant le site `https://gotofail.com/`.

Question 20. Affichez le $6^{\text{ème}}$ élément du vector `V`. Puis triez `V`. Affichez de nouveau le $6^{\text{ème}}$ élément de `V`.

```
int main() {
  std::vector<int> V {3, 1, 4, 1, 5, 9, 2, 6};

  // Votre code ici
}
```