UNIVERSITY OF HULL

500083 Revision (May)

UNIVERSITY OF HULL

```
MyClass& clip (MyClass value, MyClass low, MyClass high) {
     if (value < low)
          return low;
     if (value > high)
          return high;
     return value;
}
```

Which of the following statements best describe this piece of code?

Select **ALL** that apply

(Incorrect answers will reduce the total mark for this question, but never below zero)

A. Inefficient due to passing parameters by value

B. Unsafe because returning a reference to a location on the stack that does not exist outside the scope of the method

C. Unsafe because the parameters (value, low and high) can be changed within the method and then alter the actual parameters in the calling code.

D. None of the above

Answers = A, B

UNIVERSITY OF HULL

When is it best practice in C++ to use a virtual destructor?

A. Never

B. Always

C. When a class contains a virtual function or is a base class

D. When the class contains a friend function

Answer = C

UNIVERSITY OF HULL

```
class Fruit {
     Color _color;
public:
     Fruit(const Color &color)
          : _color(color)  { cout << "Fruit" << endl; }
};

class Apple : public Fruit {
     float _diameter;
public:
     Apple(const Color &color, float diameter)
          : _diameter(diameter), Fruit(color) { cout << "Apple" << endl; }
};
```

Which of the statements best describe the execution order for the following line of C++ code.

```
Apple anApple(red, 12);
```

A. 1. Apple constructor initialisation list; 2. Apple constructor body;
   3. Fruit constructor initialisation list; 4. Fruit constructor body
B. 1. Fruit constructor initialisation list; 2. Fruit constructor body;
   3. Apple constructor initialisation list; 4. Apple constructor body
C. 1. Apple constructor initialisation list; 2. Fruit constructor initialisation list;
   3. Fruit constructor body; 4. Apple constructor body
D. 1. Apple constructor initialisation list; 2. Fruit constructor initialisation list;
   3. Apple constructor body; 4. Fruit constructor body
E. 1. Fruit constructor initialisation list; 2. Apple constructor initialisation list;
   3. Fruit constructor body; 4. Apple constructor body

Answer = C

UNIVERSITY OF HULL

Given:

```
class Fruit { ... };
class Orange : public Fruit { ... };
```

Which of the following methods are NEVER implicitly called at position XXX in the following code?

```
Orange::Orange(const Colour &colour) XXX { .... }
```

A. `Orange::Orange()`

B. Default constructors for data member within class Orange

C. `Fruit::Fruit()`

D. Default constructors for data member within class Fruit

E. A and C

F. A and D

Answer = F

UNIVERSITY OF HULL

The following data members can be initialised in a constructor's initialisation list, but which can also be initialised within the constructor's body?

A. `int & number1;`

B. `int number2;`

C. `const int number3;`

D. B and C

E. All

Answer = B

UNIVERSITY OF HULL

Gang Of Three

Which of the following statements concerning the Gang Of Three are correct?

1. The Gang of Three consists of the default constructor, copy constructor and destructor.
2. The Gang of Three consists of the assignment operator, copy constructor and destructor.
3. We are required to implement the Gang of Three, if the class contains any data members
4. The compiler provides the Gang of Three, so we don't need to worry about it.
5. We need to explicitly implement the Gang of Three, if we wish to perform shallow copying.
6. None of the above.

Tick all that apply

Answer: 2

UNIVERSITY OF HULL

Dynamic memory

- Consider the following code:

```
1. int number;
2. cin >> number;
3. int *list = new int (number);
4. for (int i=0; i<number; i++) {
5.     cin >> list [i];
6. }
```

- Which of the following statements correctly describe the code:
  1. Fails to compile
  2. Compiles and executes correctly
  3. Exhibits a memory leak
  4. Exhibits a memory access violation
  5. Entering a negative value is potentially unsafe

3, 4

5 – its safe, since a negative number will give zero memory allocation

UNIVERSITY OF HULL

Copy constructors

- Which type of copy constructor is required for each class?

```
Class A {
   int a, b;
   ...
};
```

```
Class B {
   int list[100];
   float x;
   ...
};
```

```
Class C {
   float *table;
   short size;
   ...
};
```

```
Class D {
   int &count;
   float average;
   ...
};
```

```
Class E {
   IntList marks;
   string name;
   ...
};
```

	Class A	Class B	Class C	Class D	Class E
Shallow Copy					
Deep Copy					

Shallow A, B, E
Deep C, D - due to pointers and ref

UNIVERSITY OF HULL

Assignment operator

Spot the logical error

```
1.  class Student {
2.      string _name;
3.      int *_moduleMarks;
4.      int _numOfModules;
5.  public:
6.      Student& operator=(const Student &rhs) {
7.              _name = rhs._name;
8.              _numOfModules = rhs._numOfModules;
9.              delete[] _moduleMarks;
10.             _moduleMarks = new int[_numOfModules];
11.             for (int i = 0; i < _numOfModules; i++)
12.                     _moduleMarks[i] = rhs._moduleMarks[i];
13.             return *this;
14.     }
15.     ……
```

Missing conditional to test whether rhs == this

UNIVERSITY OF HULL

Assignment operator

Spot the logical error

```
1.  class Student {
2.      string _name;
3.      int *_moduleMarks;
4.      int _numOfModules;
5.  public:
6.      Student& operator=(const Student &rhs) {
7.              if (this != &rhs) {
8.                      _numOfModules = rhs._numOfModules;
9.                      delete[] _moduleMarks;
10.                     _moduleMarks = new int[_numOfModules];
11.                     for (int i = 0; i < _numOfModules; i++)
12.                             _moduleMarks[i] = rhs._moduleMarks[i];
13.             }
14.             return *this;
15.     }
16.     ……
```

Forgot to copy the name

UNIVERSITY OF HULL

Assignment operator

Spot the logical error

```
1.  class Student {
2.      string _name;
3.      int *_moduleMarks;
4.      int _numOfModules;
5.  public:
6.      Student& operator=(const Student &rhs) {
7.              if (this != &rhs) {
8.                      _name = rhs._name;
9.                      delete[] _moduleMarks;
10.                     _moduleMarks = new int[_numOfModules];
11.                     for (int i = 0; i < _numOfModules; i++)
12.                             _moduleMarks[i] = rhs._moduleMarks[i];
13.             }
14.             return *this;
15.     }
16.     ……
```

Forgot to initialise _numOfModules

UNIVERSITY OF HULL

Assignment operator

Spot the logical error

```
1.  class Student {
2.      string _name;
3.      int *_moduleMarks;
4.      int _numOfModules;
5.  public:
6.      Student& operator=(const Student &rhs) {
7.              if (this != &rhs) {
8.                      _name = rhs._name;
9.                      _numOfModules = rhs._numOfModules;
10.                     _moduleMarks = new int[_numOfModules];
11.                     for (int i = 0; i < _numOfModules; i++)
12.                             _moduleMarks[i] = rhs._moduleMarks[i];
13.             }
14.             return *this;
15.     }
16.     ……
```

Memory leak

UNIVERSITY OF HULL

How many method calls result from line 17?

```
1.  class Vector2f {
2.     float _x, _y;
3.  public:
4.     Vector2f();
5.     Vector2f(float x, float y);
6.     Vector2f scaleBy(const float magnitude) const;
7.  };
8.  …
9.  Vector2f Vector2f::scaleBy(const float magnitude) const {
10.    Vector2f result;
11.    result._x = _x * magnitude;
12.    result._y = _y * magnitude;
13.    return result;
14. }
15. …
16. Vector2f a(1,2), b;
17. b.scaleBy(10.0f);
```

Answer = 5
scaleBy, result (default constructor+destructor) and return by value (copy constructor+destructor),

UNIVERSITY OF HULL

Which output is correct, for lines 13 and 14?

```
1. class Fruit {
2. public:
3.    Fruit() { cout << "Fruit::Fruit()" << endl; }
4.    ~Fruit() { cout << "Fruit::~Fruit()" << endl; }
5. };
6. class Apple : public Fruit {
7. public:
8.    Apple() { cout << "Apple::Apple()" << endl; }
9.    ~Apple() { cout << "Apple::~Apple()" << endl; }
10.};
11.…
12.{
13.  Apple anApple;
14.}
```

A
Apple::Apple()
Apple::~Apple()

B
Apple::Apple()
Fruit::Fruit()
Fruit::~Fruit()
Apple::~Apple()

C
Fruit::Fruit()
Apple::Apple()
Apple::~Apple()
Fruit::~Fruit()

D
Apple::Apple()
Fruit::Fruit()
Apple::~Apple()
Fruit::~Fruit()

E
None of these

Answer = C

UNIVERSITY OF HULL

Which output is correct, for lines 13 and 14?

```
1. class Fruit {
2. public:
3.    Fruit() { cout << "Fruit::Fruit()" << endl; }
4.    ~Fruit() { cout << "Fruit::~Fruit()" << endl; }
5. };
6. …
7. class Apple : public Fruit {
8. public:
9.    Apple() { cout << "Apple::Apple()" << endl; }
10.   ~Apple() { cout << "Apple::~Apple()" << endl; }
11.};
12.…
13.Apple* ptrApple = new Apple;
14.delete ptrApple;
```

A
Apple::Apple()
Apple::~Apple()

B
Apple::Apple()
Fruit::Fruit()
Fruit::~Fruit()
Apple::~Apple()

C
Fruit::Fruit()
Apple::Apple()
Apple::~Apple()
Fruit::~Fruit()

D
Apple::Apple()
Fruit::Fruit()
Apple::~Apple()
Fruit::~Fruit()

E
None of these

Answer = C

UNIVERSITY OF HULL

Which output is correct, for lines 13 and 14?

```
1. class Fruit {
2. public:
3.    Fruit() { cout << "Fruit::Fruit()" << endl; }
4.    ~Fruit() { cout << "Fruit::~Fruit()" << endl; }
5. };
6. …
7. class Apple : public Fruit {
8. public:
9.    Apple() { cout << "Apple::Apple()" << endl; }
10.   ~Apple() { cout << "Apple::~Apple()" << endl; }
11.};
12.…
13.Fruit* ptrFruit = new Apple;
14.delete ptrFruit ;
```

A
Apple::Apple()
Apple::~Apple()

B
Apple::Apple()
Fruit::Fruit()
Fruit::~Fruit()
Apple::~Apple()

C
Fruit::Fruit()
Apple::Apple()
Apple::~Apple()
Fruit::~Fruit()

D
Apple::Apple()
Fruit::Fruit()
Apple::~Apple()
Fruit::~Fruit()

E
None of these

Answer:

Fruit::Fruit()
Apple::Apple()
Fruit::~Fruit()

Because no virtual destructor

UNIVERSITY OF HULL

Feedback from test 1

UNIVERSITY OF HULL

Parameter passing

- In terms of the data security of both the original object and any parameters, which of the following method prototypes is considered the most secure?
  1. Vector3f& vector3f::add(vector3f &rhs);
  2. Vector3f& vector3f::add(const vector3f &rhs);
  3. Vector3f vector3f::add(vector3f &rhs);
  4. vector3f vector3f::add(const vector3f &rhs);

Answer = 4

UNIVERSITY OF HULL

Arrays

Given:

```
1. short list[100];
2. short *ptrA = &list[30];
3. short *ptrB = ptrA - 10;
4. for (unsigned int i=0; i<100; i++)
5.       list [i] = i;
```

What is the value of:

list[8] + ptrA[1] + ptrB[2]

Answer = 61

UNIVERSITY OF HULL

Order of precedence

- C++ uses a strict order of precedence for implicit type conversions.
- Assign values 1 to 6 to the following types to indicate this precedence; where 1 is the most important and 6 the least important
  - Unsigned long
  - Float
  - Short
  - Int
  - Double
  - Long

UNIVERSITY OF HULL

Assembly

- Given:
  - int result;
- And:
  - call readCode (4111C7h)
  - mov dword ptr [result], eax

- Which of the following prototypes when dissassembled, best match the assembly code?
  1. int readCode(int *result);
  2. void readCode();
  3. int* readCode();
  4. int* readCode(int result);
  5. int readCode();

Answer = 5