Programming in C++: Assignment Week 7

Total Marks : 20

Right hand side of each question shows the mark and its Type (MCQ/MSQ/Programming)

Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology
Kharagpur-721302
ppd@cse.iitkgp.ernet.in

September 3, 2017

Question 1 MCQ, *Mark 1*

- Q. When a virtual function is redefined by the derived class, it is called ———

  a) Overloading
  b) Overriding
  c) Rewriting
  d) All of these

  Answer: b

Question 2 MCQ, *Mark 1*

- Q. How many parameters does a conversion operator may take ?

  a) 0
  b) 1
  c) 2
  d) Any No.of

  Answer: a

Question 3 MSQ, *Mark 1*

- Q. Look at following casting statements in the given code.

```
#include <iostream>
class Foo{ };
class Bar{ };

int main()
{
    Foo* f = new Foo;

    Bar* b2 = static_cast<Bar*>(f);           // stmnt-1
    Bar* b3 = dynamic_cast<Bar*>(f);          // stmnt-2
    Bar* b4 = reinterpret_cast<Bar*>(f);      // stmnt-3
    Bar* b5 = const_cast<Bar*>(f);            // stmnt-4

    return 0;
}
```

Identify the Incorrect casting statement/s

a) stmnt-1
b) stmnt-2
c) stmnt-3
d) stmnt-4

Answer: a, b, d

Question 4

Consider the following code segment. What will be the content of the virtual function table (VFT) for the instance of class C in correct order? *MCQ, Marks 1*

```
class A { public:
    virtual void f(int) { }
    virtual void g(double) { }
    virtual void d(char) { }
    int h(A *) { }
};

class B: public A { public:
    void f(int) { }
    virtual int h(B *) { }
};

class C: public B { public:
    void g(double) { }
    void d(char) { }
    int h(B *) { }
};
```

a. C::g(C*const, double)
   C::d(C*const, char)
   C::h(C*const, B*)
   B::f(B*const, int)

b. B::f(B*const, int)
   C::g(C*const, double)
   C::d(C*const, char)
   C::h(C*const, B*)

c. C::d(C*const, char)
   C::h(C*const, B*)
   C::g(C*const, double)
   B::f(B*const, int)

d. B::f(B*const, int)
   C::d(C*const, char)
   C::h(C*const, B*)
   C::g(C*const, double)

Answer: b)
Explanation: Virtual functions of the base class are placed first in the VFT, followed by its own virtual functions in the order of declaration in the class. Please refer to the course video lectures for more details.

Question 5

A class can be called as polymorphic if it contains *MSQ, Marks 1*

a. Virtual functions of its own

b. Virtual functions of base classes

c. Pure virtual functions

d. Member functions and Virtual functions

Answer: a), b), c), d)
Explanation: A polymorphic class should contain at least one virtual function. It can be pure, non-pure or derived with or without non-virtual functions.

Question 6 **MSQ, *Mark 1***

- Q. Look at the following code

```
#include <iostream>
using namespace std;
class A { } ;
 int main(){
     int i = 20, *x ; double d = 35.6 ;
     A a , *p = &a ;
     x = (int *) p ; // statement-1
     i = (int) d ;  // statement-2
     p = (A *) &i ; // statement-3
     return 0 ;
 }
```

Look at the C-style casting in the above code and identify the correct equivalent C++ castings associated with the statement-1, 2 &3.

a) const_cast, reinterpret_cast, static_cast

b) reinterpret_cast, const_cast, static_cast

c) static_cast, reinterpret_cast, static_cast
d) reinterpret_cast, static_cast, reinterpret_cast,

Answer: d

Question 7

Which of the following cast operator/s can convert any class object pointer to any other class object pointer? *MSQ Marks 1*

a. const_cast

b. static_cast

c. reinterpret_cast

d. dynamic_cast

Answer: c)
Explanation: reinterpret_cast. Refer course material for more details.

Question 8

MCQ, *Mark 1*

- Q. What is the return type of the conversion operator ?

  a) void *
  b) int *
  c) float *
  d) No return type

  Answer: d

Question 9 MCQ, *Mark 1*

- Q. What is the output of the following program ?

```
#include <iostream>
#include <typeinfo>
using namespace std;
class A { };
class B : public A {};
int main() {
    A a; B b ;
    A *p = &a;
    A &r1 = a;
    bool b1 = typeid(a).name() == typeid(p).name() ;
    bool b2 = typeid(a).name() == typeid(A).name() ;
    bool b3 = typeid(A).name() == typeid(p).name() ;
    bool b4 = typeid(r1).name() == typeid(a).name() ;
    p = &b ;
    bool b5 = typeid(p).name() == typeid(b).name() ;
    bool b6 = typeid(p).name() == typeid(B).name() ;
    bool b7 = typeid(p).name() == typeid(A).name() ;
    cout << b1 << b2 << b3 << b4 <<b5 << b6 << b7 ;
    return 0;
}
```

a) 1101001
b) 1101010
c) 0101000
d) 0101001

Answer: c

Question 10 MCQ, *Mark 1*

- Q. Look at the code bellow, Replace the `line-1` by a correct casting statement (if required).

```
class A { };
class B: public A { };
...........
A a ;
B *p = static_cast<B*>(&a) ; // Line-1
```

a) B *p = const_cast <B*> (&a) ;
b) B *p = reinterpret_cast <B*> (&a) ;
c) B *p = dynaqmic_cast <B*> (&a) ;
d) No changes required

Answer: c

Question 1 Programming, *Mark 3*

- **Problem statement:** Look at the Following program
  Note: Write proper constructor/conversion operator in appropriate class to make function convert( ) workable, which cast/convert 'unrelat' to 'unrelat_1' class type

```
------------------------- Prefixed Fixed Code --------------------------
#include <iostream>
using namespace std;
------------------------- Template Code (Editable) ----------------------
class unrelat{};
class unrelat_1 { };
-------------------------- Suffixed Fixed Code --------------------------
int Convert( class unrelat_1 &c, class unrelat &d){
    int i = 5;
    cin >> i ;
    c = d;
    c = static_cast<unrelat_1>(d);
    c = (unrelat_1)d;
    return (i*10+35) ;
}
int main() {
    int out;
    unrelat_1 a;
    unrelat b;
    out = Convert(a, b);
    cout << out ;
    return 0;
}
-------------------------------------------------------------------
```

Public

```
Input: 2
Output: 55
Input: 15
Output: 185
```

Private

```
Input: 1
Output: 45
```

Answer:

```
class unrelat_1 {
public:
    unrelat_1(){  }
    unrelat_1(const unrelat&) {  }
};
```

```
Answer: --------------OR------------------
class unrelat{
public:
    operator unrelat_1()
        { return unrelat_1(); }
};
```

Question 2 MSQ, *Mark 4*

- **Problem Statement:** Write the appropriate constructor/operator function in following Program To cast 'unrelat' class to char * and vice versa
  Note: Except class content, other section of code is not editable. write proper constructor/conversion operator function in appropriate class to make function convert() workable.

```
------------------------- Prefixed Fixed Code --------------------------
#include <iostream>
#include<cstring>
using namespace std;
------------------------- Template Code (Editable) ---------------------
class unrelat { public: char  *str ;

// Write the appropriate constructor and/or
// conversion operator function  here, as per the requirement to make convert( ) wo

};
----------------------- Suffixed Fixed Code --------------------------
char * Convert( class unrelat &a, char * i){
    char s[20] ;
    // unrelat ==> char *
    i = static_cast<char*>(a);
    i = (char *)a;
    strcat(i,"-success-");
    cin >> s ;
    // char * ==> unrelat
    a = static_cast<unrelat>(s);
    a = (unrelat)s;
    strcat(i,a.str);
    return (i) ;
}
int main() {
    char * out;
    char input[20] ;
    cin >> input ;
    unrelat x (input);
    out = Convert(x, input);
    cout << out ;
    return 0;
}
---------------------------------------------------------------------
```

- **Answer:**

```
    unrelat(char * str_= 0): str(str_) { }
    operator char *()
        { return (str); }
```

Public

```
Input: my
       string
Output: my-success-string
Input: what
       is
Output: what-success-is
```

Private

```
Input: cast
       test
Output: cast-success-test
```

Explanation: To convert Built-in type to class type we require a proper constructor (Not Operator function). Similarly To convert (cast) a class type to Built-in type we require a proper operator function (Not constructor).

Question 7 **Programming, *Mark 3***

- **Problem statement:** Consider the following program. Write proper constructor / conversion operator as per the requirement in the editable section to match the outputs of test cases.

```
#include <iostream>
using namespace std;

class A {
    int i;
public:
    A(int ai) : i(ai) {}
    int get() const { return i; }
    void update() { ++i; }
};

class B {
    int i;
public:
    B(int ai) : i(ai) {}
    int get() const { return i; }

    //------------------------ Template Code (Editable) --------------------
    -----------------------------

    -----------------------------
    //------------------------- Suffix Fixed Code --------------------------
    void update() { ++i; }
};
int main() {
    int i;
    cin >> i;

    A a(i++);
    B b(i);

    const B &r = static_cast<B>(a);
    a.update();
    cout << a.get() << ":";
    cout << r.get() << ":";

    const A &s = static_cast<A>(b);
    b.update();
    cout << b.get() << ":";
    cout << s.get() << ":";

    return 0;
}
//--------------------------------------------------------------------
```

Public 1

```
Input:
1
Output: 2:1:3:2:
```

Public 2

```
Input:
15
Output: 16:15:17:16:
```

Private

```
Input:
5
Output: 6:5:7:6:
```

Answer:

```
    B(A& a) : i(a.get()) {}

    operator A() { return A(i); }
```

Explanation: `static_cast` can explicitly call a single-argument constructor or a conversion operator (that is, User-Defined Cast) to handle the casting between two unrelated classes. Here both are present.