Data Structures and Object-oriented Programming
Instructor: Prof. Liwei Chan
Quiz1 Date:2021/04/19

Student ID:________________ Student Name:

Part I (40%, 20% each)

1. Which of the following statements is **Incorrect**?
   a. A class can have a virtual constructor.
   b. A class is abstract if it has at least one pure virtual function.
   c. If you created a virtual function in the base class and it is being overridden in the derived class then we don’t need a virtual keyword in the derived class.
   d. Virtual functions should be accessed using pointers or references of base class type to achieve run-time polymorphism.

2. What is the output of the following code?

```
void foo(int &a, int b, int *c){
   a *= 2;
   b *= 2;
   *c *= 2;
}

int main(){
   int x = 1, y = 3, z = 7;
   foo(x, y, &z);
   cout << x << ", " << y << ", " << z << endl;
   return 0;
}
```

   a. 2, 6, 14
   b. 1, 3, 7
   c. 2, 3, 14
   d. 1, 6, 7

Part II (60%, 30% each)

1. Does the following code have error(s)? If there is, which type of errors is(are) it, run-time or compilation time error? And please indicate the problem(s) of this code. If there is no error, then what is the output?

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

int main() {
    int a=5, b=10;
    int* p1, p2;
    p1 = &a;
    p2 = &b;
    *p1 = 2;
    cout << a<< " " << b << endl;

    return 0;
}
```

```
Output:
compilation time error
p2是int, 不是int*
```

2. Please draw the graph to display the value of c, p1 and p2. A pointer can use "→" to represent where it points to, if the value is unknown please write "?".

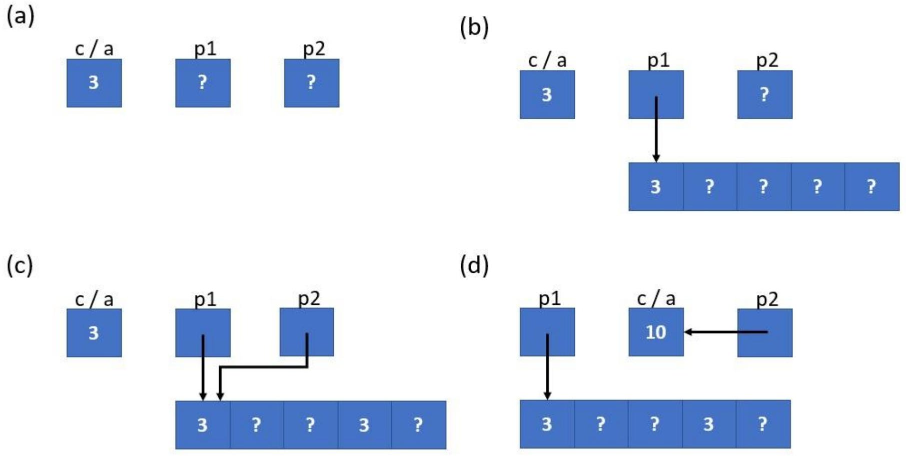
```
int main(){
    int a = 3;
    int b = 5;
    int &c = a;
    int *p1;
    int *p2;
    // (a) state 1

    p1 = new int[b];
    *p1 = c;
    // (b) state 2

    p2 = p1;
    *(p2 + a) = c;
    // (c) state 3

    p2 = &c;
    *p2 = 10;
    // (d) state 4
    delete [] p1;
    return 0;
}
```

answer:



各小題都要寫出c/p1/p2三者的value(除非其狀態沒改變),且錯一個全扣:
p1,p2必須用箭頭畫出指著的位置
a)的p1, p2應為空白或標?

b,c,d)的動態分配的array不會初始化， 不能填0

有幾位同學把state狀態往下看了,那樣的d題:p1指向的位置不會改變，但是由於空間可能會被重新分配出去，所以值是未知的(dangling pointer)