Inheritance

Problem 1: What is the output for the following program?

```
#include <iostream>
using namespace std;
class A
{
public:
        void print(){cout << “A::print()"<<endl;}
};

class B : public A
{
public:
        void print(){cout << "B::print()"<<endl;}
};

int main()
{
        A a;
        a.print();
        B b;
        b.print();
        A* c = new B;//Polymorphism
        c->print();
}
```

What if we add virtual keyword to print() in class A?

Problem 2: What’s the output?

```
#include <iostream>
using namespace std;
class A
{
public:
    virtual void print(){cout << "A::print()"<<endl;}
    virtual ~A(){};
};

class B:public A
{
public:
    void print(){cout << "B::print()"<<endl;}
};

void printSomething_1(A a){ a.print();}
void printSomething_2(A& a){ a.print();}
void printSomething_3(A* a){ a->print();}
```

```
int main() {
    B b;
    A* c = new B;
    printSomething_1(b);
    printSomething_2(b);
    printSomething_3(&b);
    printSomething_1(*c);
    printSomething_2(*c);
    printSomething_3(c);
}
```

What if print() in class A is a pure virtual function?

Problem 3: What’s the output?

```
#include <iostream>
using namespace std;
class A
{
public:
    A(){cout << "A()" << endl;}
    A(int x){cout<< "A(" << x << ")" << endl;}
    ~A(){cout << "~A()" << endl;}
};

class B
{
public:
    B(){cout << "B()" << endl;}
    B(int x):m_a(x){cout << "B(" << x << ")" << endl;}
    ~B() {cout << "~B()" << endl;}
private:
    A m_a;
};

class C:public A
{
public:
    C():A(10), m_b2(5){ cout << "C()" << endl;}
    ~C(){ cout << "~C()"<< endl;}
private:
    B m_b1;
    B m_b2;
};

int main() {
    C c;
}
```

Recursion

Problem 1: Return the factorial of n using recursion

```
// Assume n is a nonnegative integer
int fact(int n)
{






}
```

Problem 2: Print the elements of an array in order

```
void printArrayInOrder(int arr[], int n)
{






}
```

How about printing the array in reverse order?

Problem 3: Return a^b

```
// Assume b is a nonnegative integer
int expon(int a, int b)
{






}
```

Problem 4: Return fibonacci(n)

```
// Assume b is a nonnegative integer
int fab(int n)
{






}
```

Problem 5: How many ways are there for you to go n steps?

```
// You can go either 1 or 2 steps each time.
int step(int n)
{

}
```

Problem 6: Parade organization

```
//You are asked to organize a parade consisting of bands and floats.
//You can't place a band immediately after another.
//How many ways can you organize a parade of size n?
//i.e. for n = 2, you have 3 ways to organize the parade: float-float,
float-band and band-float.

void solveParade(int n)
{

}
```