

Introduction to Programming Systems Design

CSCI 455 SAMPLE FINAL SOLUTION

Dr. K. Narayanaswamy
Final Examination

You have 2 hours (120 minutes) to complete this examination. Cell phones/laptops not allowed. You can use any and all printed reference materials – however, you will NOT be allowed to share any reference materials, including books, notes, etc. Each student must have his/her own reference materials.

YOU MUST ANSWER **all** questions in both parts. Part I contains mostly objective questions worth 1 point each **unless otherwise specified**. Part II contains 6 questions, each worth 11 points.

NAME: _____

STUDENT ID: _____

| QUESTION | MAXIMUM | SCORE |
|----------------|--------------|-------|
| PART I | 28 | |
| PART II | | |
| Question 1 | 12 | |
| Question 2 | 12 | |
| Question 3 | 12 | |
| Question 4 | 12 | |
| Question 5 | 12 | |
| Question 6 | 12 | |
| TOTAL | (100) | |

PART I: Clearly MARK your answer. Ambiguous notations will not be given credit. Each question carries 1 point, unless otherwise specified.

- 1) The construct that supports inheritance in C++ is called:
1) function 2) enum 3) typedef **4) class**
- 2) By default, all members of a class are _____ and all members of a struct are _____:
1) public, public 2) public, private
3) private, public 4) private, private
- 3) Access to private functions of a class is available to:
1) only public members of class 2) any place the class is visible
3) only private member functions **4) all member functions**
- 4) If a class provides a single constructor function, there are conditions under which it might not be called when a new object is created:
TRUE / **FALSE**
- 5) Constructor function in C++:
1) May return void 2) Cannot take parameters
3) May return an object **4) Cannot return any value**
- 6) Destructor function in C++ cannot:
1) Delete any attributes **2) Be overloaded**
3) Delete protected members 4) Be defined inside the class
- 7) The most common kind of derivation in C++ programs as a whole is:
1) Protected derivation 2) Private derivation
3) Public derivation 4) Answer is impossible
- 8) When is the base class constructor called relative to the derived class constructor?
1) They are called together 2) Depends on kind of derivation
3) Derived class constructor first **4) Base class constructor first**
- 9) When the compiler cannot find a member function in a class for a particular object, it will look for:
1) Any function matching that name 2) Least specific member function
3) Least general function possible 4) Any function that looks available

```
class Polygon
{
    public:
        Polygon (double );
        ~Polygon ();
        Polygon (const Polygon &p);
        void print () const;
```

```
protected:
    double area;
private:
    int id;
};
```

10) The function Polygon (double) is a class: [2 points]

- 1) constructor
- 2) destructor
- 3) copy constructor
- 4) none of these

11) Given the definition: Polygon p1(1.1); what values are passed to the constructor? [2 points]

- 1) 0.0
- 2) 1.1
- 3) area
- 4) none of these

12) Assume p1 of type Polygon in #11 is an automatic local variable. When the variable p1 goes out of scope, which function is automatically called? [2 points]

- 1) Polygon (double)
- 2) ~Polygon ()
- 3) Polygon (const Polygon &p)
- 4) none of these

13) Which function can access the data member area of p1?

- 1) Members of all classes
- 2) Only Members of Polygon
- 3) Only Members of Polygon and its subclasses
- 4) none of these

```
class Triangle : public Polygon
{
    public:
        Triangle ();
        ~Triangle ();
    private:
        Point p1;
        Point p2;
        Point p3;
};
```

14) Triangle's use of class Polygon is an example of

- 1) Composition
- 2) Inheritance
- 3) Dynamic binding
- 4) None of these

15) Triangle's use of class Point is an example of

- 1) Composition
- 2) Inheritance
- 3) Dynamic binding
- 4) None of these

16) The class methods of Triangle can access the private members of Polygon.

TRUE / FALSE

17) The private data members of a class are part of its:

- 1) Interface
- 2) Base class
- 3) Implementation
- 4) Protected members

18) If I wish to allow public base class functions to be used via derived class objects in the “main” function, I must use:

- 1) private derivation
- 2) public derivation
- 3) protected or public derivations
- 4) protected derivation

19) If I need to do hundreds of additions to the front of a collection, a few additions to the end of a collection, and none in the middle, the ideal STL collection is:

- 1) list
- 2) vector
- 3) None of the above
- 4) deque

20) Which of the following is not appropriate as an indexed collection:

- 1) list
- 2) vector
- 3) All of the above
- 4) deque

21) An STL vector is better than an ordinary C++ array because:

- 1) Vector can grow as needed
- 2) Vector supports random additions
- 3) Vector is indexed
- 4) Vector is more efficient

22) Say that a variable X is declared as below:

```
vector<int> X;
```

X.size() can never be larger than X.capacity(). Explain briefly.

Size() is the actual number of elements in the vector; capacity() is the number of elements above which the vector has to be resized. Therefore size() <= capacity() at all times.

23) Say you are designing a class C. X is a member of that class that you feel should not be used outside the class. What access level would you select for X?

- 1) public
- 2) protected
- 3) None of the above
- 4) private

24) Say you are designing a class C. Y is a member of that class that is required to use objects of type C. What access level would you select for Y?

- 1) public
- 2) protected
- 3) None of the above
- 4) private

25) Say you are designing a class C. Z is a member of that class that is required to extend the class C. What access level would you select for Z?

- 1) public
- 2) protected
- 3) None of the above
- 4) private

27) **(3 points)** Write down the entire definition of the function “printStringStatistics” based on the following description and example output. You may assume access to standard C/C++ libraries. The function returns no values, but prints out the following:

- a) The total number of uppercase alphabetic letters in the string
- b) The total number of lowercase alphabetic letters in the string
- c) The number of numeric digits in the string
- d) The number of other characters (not in either category a, b, or c above)

You may use any library functions available in the standard C/C++ libraries.

For example,

```
printStringStatistics("$Jus&tin123")
```

yields as output:

```
Uppercase characters: 1  
Lowercase characters: 5  
Digits: 3  
Others: 2
```

```
void printStringStatistics (string input)  
{  
    int lccount = 0;  
    int uccount = 0;  
    int dcount = 0;  
    int ocount = 0;  
    for(int i = 0; i < input.size(); i++)  
    {  
        if (isupper(input[i]))  
            uccount++;  
        else (islower(input[i]))  
            lccount++;  
        else if (isdigit(input[i]))  
            dcount++;  
        else  
            ocount++;  
    }  
    cout << "Uppercase characters = " << uccount << endl;  
    cout << "Lowercase characters = " << lccount << endl;  
    cout << "Digits = " << dcount << endl;  
    cout << "Others = " << ocount << endl;  
}
```

PART II: Question 1: (12 points) Answer the questions related to the following code:

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

int itemCount = 20;
class Item
{
    public:
        int get_weight () { return weight; }
        void set_weight (int w) { weight = w; }
        Item (int w) { weight=w ; itemCount--;}
        Item () { weight=0; itemCount--;}
        ~Item () { itemCount++; }
    protected:
        int weight;
};

main ()
{
    Item a(5), b(9), arr[7];
    cout << "Count of items: " << itemCount << endl ;           // LINE 1
    Item *p = new Item();
    {
        Item c(16), arr2[5];
        p = new Item(31);
        cout << "Count of items: " << itemCount << endl ;       // LINE 2
    }
    delete p;
    cout << "Count of items: " << itemCount << endl ;           // LINE 3
}
```

a) Briefly explain why it is not legal for the following statement to appear in the main program after LINE 1 (2 points)

```
cout << arr[0].weight ;
```

weight is protected member of Item, and cannot be accessed directly by main, which is a non-member function.

b) What is the output produced by LINE 1 of the main program above? (3 points)

Count of items: 11

c) What is the output produced by LINE 2 of the main program above? (4 points)

Count of items: 3

d) What is the output produced by LINE 3 of the main program above? (4 points)

Count of items: 10

PART II: Question 2: (12 points) Answer the questions related to the following code:

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

class Part
{
    public:
        Part (int s) { baseSpeed = s; }
        int getSpeed () { return baseSpeed; }
    private:
        int baseSpeed;
};

class CPU : public Part
{
    public:
        CPU (int s, int t) : Part (s) { turbo = t; }
        int getSpeed () { return Part::getSpeed() + (2 * turbo); }
    private:
        int turbo;
};

main ()
{
    Part *p;
    CPU C(10, 3);
    cout << C.getSpeed() << endl;           // Line # 1
    p = new CPU(20, 2);
    cout << p->getSpeed() << endl;       // Line # 2
}
```

a) What is the output of Line # 1 ? (2 points)

16

b) What is the output of Line # 2 ? (2 points)

20

Supposing we add the word "virtual" from the declaration of "get_string" in the class Part. **In the changed program**, answer the following 2 questions:

a) What is the output of Line # 1 ? (4 points)

16

b) What is the output of Line # 2 ? (4 points)

24

PART II: Question 3. (12 points)

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

// represents a collection of job numbers, which are integers
class jobList {
public:
    void addJob (int job) { if (! jobExists(job)) db.push_back(job);
                          else cout << "Duplicate job not added" << endl; }
    bool jobExists (int job); // return true if job exists in db, false otherwise
    void deleteJob (int job); // remove job if it exists in db, error msg otherwise

private:
    list<int> db;
};
```

a) Complete the body of the function “jobExists” shown below: (2 points)

```
bool jobList::jobExists (int job)
{
    // answer 1:
    return find(db.begin(), db.end(), job) != db.end() ;

    // answer 2:
    if (find(db.begin(), db.end(), x) != db.end()) return true;
    else return false;

    // answer 3:
    for (list<int>::iterator p = db.begin(); p != db.end(); p++)
        if (*p == job) return true;
    return false;
}
```

b) Complete the body of the function “deleteJob” shown below (print out an error message if the function is instructed to delete a job that does not currently in the job list) (2 points). You may use “jobExists” function IF you wish, but this is not a requirement.

```
void jobList::deleteJob (int job)
{
    list<int>::iterator p = find(db.begin(), db.end(), job);
    if (p != db.end())
        db.erase(p);
    else
        cout << job << " cannot be deleted" << endl;
}
```

c) The current implementation of the member function “addJob” adds job numbers to the end of the list. Show how you would change this function **to maintain the job numbers in sorted (ascending) order**, so that lower job numbers always occur before the larger numbers. (8 points)

```
bool job_list::addJob (int job)
{
    list<int>::iterator p = db.begin();

    while (p != db.end())
    {
        if (*p < job) p++;
        else if (*p == job)
        {
            cout << "Duplicate. Not added" << endl;
            return;
        }
        else
        {
            db.insert(p, job);
            return;
        }
    }
    db.push_back(job);
}
```

PART II: Question 4. (12 points)

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

int mystery (int array[], int n)
{
    if ( n <= 0) return 0;
    return array[n] + mystery(array, n-1) - mystery(array, n-2);
}

main ( )
{
    const int size = 6;
    int collection1[size], collection2[size];
    for (int i = 0; i < size; i++)
        collection1[i] = i * i;
    for (int i = 0; i < size; i++)
        cout << collection1[i] << " ";           // Line 1
    cout << endl;
    for (int i = 0; i < size; i++)
        collection2[i] = mystery(collection1, i);
    for (int i = 0; i < size; i++)
        cout << collection2[i] << " ";           // Line 2

    cout << endl;
}
```

a) What is the entire output produced by Line 1 of the program above? (3 points)

0 1 4 9 16 25

b) What is the output of Line 2 of the program above? (9 points)

0 1 5 13 24 36

PART II: Question 5. (12 points)

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

void print (int i) { cout << " " << i ; }

bool predicate1 (int i) { return i > 23 ? true : false ; }

bool predicate2 (int i) { if ( ( i % 8 ) == 0 ) return true; else return false; }

main ()
{
    list<int> foo;
    list<int>:: iterator fp1, fp2, fp3;

    for (int i = 1; i <= 5; i++) foo.push_front(i * i);
    for (int i = 6; i <= 10; i++) foo.push_front(i * 2);

    for_each(foo.begin(), foo.end(), print);           // LINE 1
    cout << endl;

    fp1 = find_if (foo.begin(), foo.end(), predicate1);
    for_each(fp1, foo.end(), print);                   // LINE 2
    cout << endl;

    cout << "First: " << count_if (fp1, foo.end(), predicate2); // LINE 3
    cout << endl;

    cout << "Second: " << count(foo.begin(), foo.end(), 16) << endl; //LINE 4

    fp3 = find_if (foo.begin(), foo.end(), predicate2);
    foo.erase(fp3, fp1);
    for_each(foo.begin(), foo.end(), print);           // LINE 5
    cout << endl;
}
```

a) What is the output of LINE 1 of the program as shown (2 points)

20 18 16 14 12 25 16 9 4 1

b) What is the output of LINE 2 of the program as shown (2 points)

25 16 9 4 1

c) What is the output of LINE 3 of the program as shown (**3 points**)

First: 1

d) What is the output of LINE 4 of the program as shown (**2 points**)

Second: 2

e) What is the output of LINE 4 of the program as shown (**3 points**)

20 18 25 16 9 4 1

PART II: Question 6. (12 points)

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

class mystery
{
public:
    mystery (int x, int y) { first = x; second = y; }
    bool operator () (int a)
    {
        if ( a >= first && a <= second ) return false;
        else return true;
    }
private:
    int first;
    int second;
};

void show (int x) { cout << " " << x; }

main ()
{
    vector<int> foo;
    for (int x = 0; x <= 4; x++) foo.push_back(x * 2);
    for (int x = 9; x >= 5; x--) foo.push_back(x * x);

    for_each(foo.begin(), foo.end(), show);//=== LINE 1
    cout << endl;

    cout << "Count 1: " << count_if(foo.begin(), foo.end(), mystery(4,80)); // LINE 2
    cout << endl;

    cout << "Count 2: " << count_if(foo.begin(), foo.end(), mystery(80, 4)); // LINE 3
    cout << endl;
    cout << "Count 3: " << count_if(foo.begin(), foo.end(), mystery(25,25)); // LINE 4
    cout << endl;

}
```

a) LINE 1 output (3 points)

0 2 4 6 8 81 64 49 36 25

b) LINE 2 output (3 points)

Count 1: 3

c) LINE 3 output (3 points)

Count 2: 10

d) LINE 4 output (3 points)

Count 3: 9