國立臺北科技大學

九十三學年度資訊工程系碩士班入學考試

軟體設計試題

填准考證號碼

注意事項：

1. 本試題共五題，配分共 100 分。
2. 請按順序標明題號作答，不必抄題。
3. 全部答案均須答在答案卷之答案欄內，否則不予計分。

Problem 1 [20%]

A programmer is designing a real-time kernel, where several processes with different priorities may be executed simultaneously. He decides that he needs a *priority queue* data structure to facilitate the scheduling of processes. His priority queue requires three operations: *insert* (insert a new process with a given priority to the queue), *remove* (remove the process with the highest priority from the queue), and *top* (return the process with the highest priority). He discovers that an array based implementation of (binary) *heap* is efficient for his priority queue. However, to minimize memory usage, he can not afford to allocate a large array. He prefers to grow and shrink his priority queue dynamically. He is asking for your help. Please answer the following questions for him.

(1-1) [3%] What are the complexities of array based heap implementation for the insert, remove, and top operations?

(1-2) [7%] In terms of complexity, discuss the advantages and disadvantages of using a binary search tree to implement a priority queue. Does it make sense to use balanced trees (e.g., AVL trees) to implement a priority queue?

(1-3) [10%] A heap can also be implemented with "real" binary tree (trees constructed by pointers and nodes) instead of array. This implementation allows the heap to grow dynamically. Suppose there are already N elements in such a heap. Please describe how to insert a new element into the heap (please describe clearly how to the find the correct initial location for the new element).

Problem 2 [20%]

Given the following C++ code, please write the output of statement 2-1 ~ 2-10.

```
class Data {
  public:
    int a;
    static int count;

    Data( int x ): a(x) {
        count++;
    }

    Data( Data& d ) {
        a = d.a;
        count++;
    }

    Data& operator = ( Data& d ) {
        a = d.a;
        return *this;
    }

    void triple() {
        a *= 3;
    }

    void print( X& x ) {
        x.printValue( *this );
    }
};

int Data::count = 0;

class X{
  public:
    virtual void printValue( Data& d ) = 0;
};

class Y : public X{
  public:
    void printValue( Data& d ) {
        cout << "Data = " << d.a << endl;
    }
};

class Z : public X
{
  public:
    void printValue( Data& d ) {
        cout<<"Count = "<<d.count<<endl;
    }

};
```

```
void main()
{
    Y y;
    Z z;

    Data m(5);
    Data n = m;
    Data& r = m;
    Data* p = &n;

    m.print(y);     //-- (2-1)[2%]
    m.print(z);     //-- (2-2)[2%]

    m.triple();
    m.print(y);     //-- (2-3)[2%]
    r.print(y);     //-- (2-4)[2%]
    n.print(y);     //-- (2-5)[2%]
    p->print(y);    //-- (2-6)[2%]

    p = new Data(7);
    r = *p;
    m.print(y);     //-- (2-7)[2%]
    n.print(z);     //-- (2-8)[2%]

    r.triple();
    r.print(y);     //-- (2-9)[2%]
    p->print(y);    //- (2-10)[2%]

    delete p;
}
```

注意：背面尚有試題

Problem 5 [20%]

Consider a proxy design pattern applied to hide the communication complexity of a networked services system as shown in Figure 5-a. In this design, a service proxy provides a local representative for the remote services in different address space via a socket connection. Given a class diagram depicting the common service interface shared by the service object and the service proxy in Figure 5-b, please write down the code to fulfill the following requirements.

(5-1) [5%] Please design a service object named **FactorialService** which contains a function call, **factorial(n)**, to perform n! calculation RECURSIVELY.

(5-2) [8%] Using socket API in Table 5, please design a socket server named **FactorialServer** which entitles FactorialService to be Internet accessible.

(5-3) [7%] Using socket API in Table 5, please design a service proxy named **ServiceProxy** which takes service calls of a client and passes them to the server.

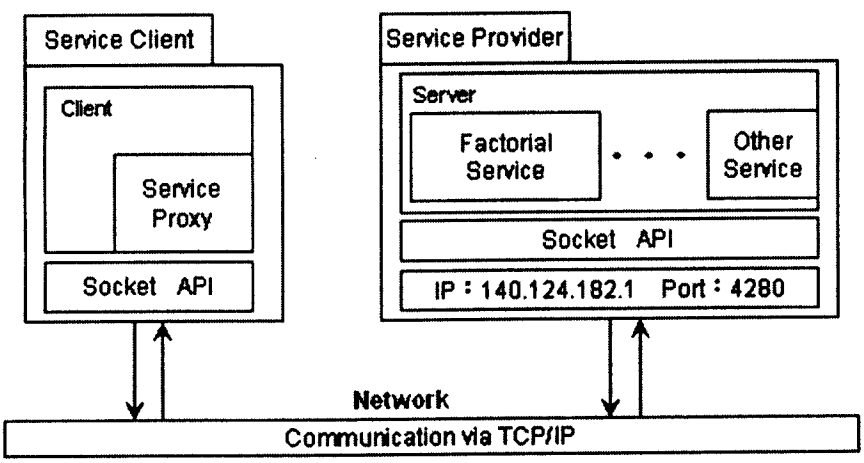

Figure 5-a　A Networked Services System

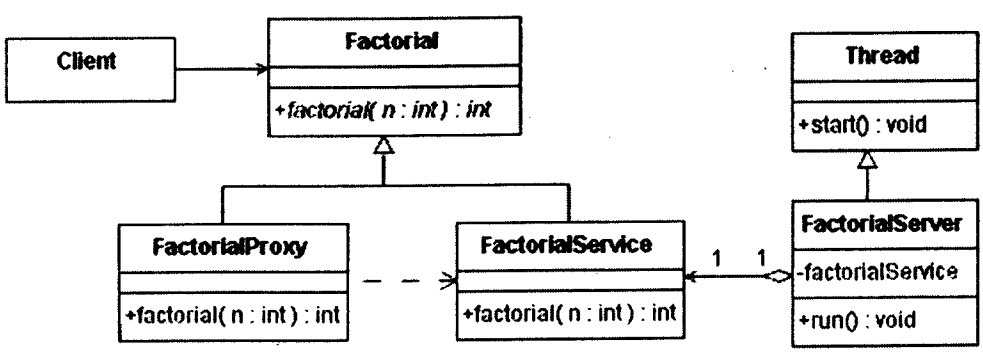

Figure 5-b　Class Diagram

Problem 4 [20%]

In a computer network, it is useful to design more than one route between the nodes of the network, so as to handle possible failures at the connection nodes. An undirected graph (computer network) is *biconnected* if and only if there are at least two different paths connecting each pair of vertices (nodes). Therefore it is useful to design computer networks as biconnected networks.

(4-1) [5%] Please give a simple graph that is biconnected and another simple graph that is not biconnected.

(4-2) [5%] An *articulation point* in a connected graph is a vertex, if deleted, would break the graph into two or more pieces. Please show that a graph is biconnected if and only if it has no articulation points.

(4-3) [5%] Articulation points can be found easily with DFS (depth-first search). Please describe the condition that a vertex x is an articulation point during DFS traversal.

(4-4) [5%] Please give an algorithm to determine if a given graph $G=(V,E)$ is biconnected.

Problem 3 [20%]

Given a stack class, a main program and its expected output below in C++, please write down the code for Problem 3-1 and Problem 3-2.

```
class Stack {

private:
   vector<Object*> stack;

public:
   void push( Object *obj ){
      stack.push_back(obj);
   }

   Object* pop(){
      if ( stack.empty() )
         return 0;
      Object* obj=stack[stack.size()-1];
      stack.pop_back();
      return obj;
   }

   bool isEmpty(){
      return stack.empty();
   }
};
```

```
void main() {

  Stack productStack, userStack;

  productStack.push(new Product(100));
  productStack.push(new Product(200));
  while ( !productStack.isEmpty() ) {
       Object* obj = productStack.pop();
       obj->print();
       delete obj;
  }

  userStack.push(new User("Alice"));
  userStack.push(new User("Bob"));
  while ( !userStack.isEmpty() ) {
       Object* obj= userStack.pop();
       obj->print();
       delete obj;
  }

}
```

```
Price is 200
Price is 100
Name is Bob
Name is Alice
```

(3-1) [10%] With polymorphism, please define a base class **Object** and implement its derived classes **Product** and **User** in C++.

(3-2) [10%] With template mechanism, please rewrite **Stack** class such that its data member can be other types.

Table 5 Socket API

■ Socket

Socket(String host, int port) Creates a stream socket and connects it to the specified port number on the host.
void close() Closes this socket.
InputStream getInputStream() Returns an input stream for this socket.
OutputStream getOutputStream() Returns an output stream for this socket.

■ ServerSocket

ServerSocket(int port) Creates a server socket on a specified port.
Socket accept() Listens for a connection to be made to this socket and accepts it.
void close() Closes this socket.

■ OutputInteger

OutputInteger(OutputStream out) Create a new OutputInteger from an existing OutputStream.
void send(int n) Send an integer.

■ InputInteger

InputInteger(InputStream in) Create an InputInteger from an existing InputStream.
int receive() Read an integer.

■ Thread

Thread() Allocates a new Thread object.
void start() Causes this thread to begin execution.
void run() The Java Virtual Machine calls the run method of this thread.