106 CI 02

國立臺北科技大學106學年度碩士班招生考試

系所組別：2300 資訊工程系碩士班

第二節　程式設計　試題

注意事項：

1. 本試題共八題，共100分。
2. 請標明大題、子題編號作答，不必抄題。
3. 全部答案均須在答案卷之答案欄內作答，否則不予計分。

Problem 1 [8%, each 2%]

Please give the best asymptotic running time for each of the problems or bound for each of the recurrences shown below using the "big oh" notation. It is assumed that $T(1)=d$ for some constant d and c is a constant in all the recurrences. Just state the answer - you **do NOT** need to justify them.

(1) $T(n)=T(n/2)+c \log n$

(2) $T(n)=T(n-1)+1/n$

(3) Finding the median in an unsorted set of size n.

(4) Performing a breadth-first search on a graph $G=(V, E)$ where $|V|=n$ and $|E|=m$.

Problem 2 [8%, each 2%]

Mark by **T** (=true) or **F** (=false) each of the following:

(1) If a problem is NP-complete, this implies that such a problem is no solution at all.

(2) There are no graphs for which Prim's algorithm can run faster than Kruskal's algorithm.

(3) Suppose problem P_1 can be reduced to problem P_2 in linear time. Then, if P_2 is NP-hard then P_1 is NP-hard.

(4) The best asymptotic running time for determining the shortest path between a given pair c vertices in a directed graph with positive weights is O(m log n), given $|V|=n$ and $|E|=m$.

Problem 3 [8%, each 4%]

Answer the following questions about heap-based priority queues. Assume a maximum-oriented priority queue.

(1) Please give the two major algorithms to fix the heap: top-down and bottom-up *heapifying*. Your algorithms should run in O(log n) time.

(2) Give the algorithms that implement the two major operations of a heap-based priority queue: insert, and remove the maximum.

Problem 4 [6%]

Let T be a binary tree rooted at r with vertex set V and edge set E. Suppose it is represented using adjacency list format. If node u is an ancestor of v, there is a path from r to v passing through u. Consider the function *ancestor*(u, v) which returns TRUE if u is an ancestor of v and FALSE otherwise. In order to have this function run in O(1) time, we are asked to design an algorithm to preprocess the tree. Please provide a linear time, i.e., O($|V|+|E|$) time, algorithm for this preprocessing.

Problem 5 [30%, each 3%]

Given the program below in C. Please trace the program and fill the 5-1~5-10 blanks with the printf output of each statement.

Problem	Answer
5-1	
5-2	
5-3	
5-4	
5-5	
5-6	
5-7	
5-8	
5-9	
5-10	

Please copy the above answer table to your answer sheet.

注意：背面尚有試題

```
#include<stdio.h>
#include<string.h>
int test01(int n){
        if (n <= 1) return 1;
        else return n * test01(n - 1);
}
int test02(){
        enum {CLUBS, DIAMONDS,
HEARTS, SPADES} s;
        int i = DIAMONDS;
        s = HEARTS;
        s++;
        i = i + s + SPADES;
        return i;
}
int test03(int n){
        int rem;
        do {
                n /= 10;
                rem = n % 10;
                if (rem != 0) break;
        } while (n > 0);
        return rem;
}
int test04(){
        char str1[10], str2[10];
        strcpy(str1, "abc");
        strcpy(str2, "abc");
        strcat(str1, strcat(str2, "ghi"));
        return strcmp(str1, str2);
}
int test05(){
        int a[] = {1, 2, 3, 4, 5};
        int *p, *q;
        p = &a[8];
        q = p - 3;
        p -= 6;
        return p - q;
}
int test06(){
        int v = 0xFF;
        v &= 1;
        v |= 0;
        v <<= 3;
        v >>= 1;
        return v;
}
```

```
int test07(){
        int grade = 4, ans = 0;
        switch (grade) {
            case 4:   ans += 4;
            case 3:   ans += 3;
            case 2:   ans += 2;
            case 1:   ans += 1;
            default: ans = 0;
        }
        return ans > 0 ? ans : 0;
}
int test08(){
        int i = 0;
        return i++;
}
int test09(int n){
        int i = 1;
        while (i < n) i *= 2;
        return i;
}
int test10(int n){
        int div;
        if (n <= 1)return 0;
        for (div = 2; div * div <= n; div++)
                if (n % div == 0) return 0;
        return 1;
}
int main(){
        printf("%d\n", test01(3));      // Problem 5-1
        printf("%d\n", test02());       // Problem 5-2
        printf("%d\n", test03(123)); // Problem 5-3
        printf("%d\n", test04());       // Problem 5-4
        printf("%d\n", test05());       // Problem 5-5
        printf("%d\n", test06());       // Problem 5-6
        printf("%d\n", test07());       // Problem 5-7
        printf("%d\n", test08());       // Problem 5-8
        printf("%d\n", test09(100)); // Problem 5-9
        printf("%d\n",test10(7));      // Problem 5-10
        return 0;
}
```

Problem 6 [18%, each 3%]

Given the program below in C++. Please trace the program and fill the 6-1~6-6 blanks with the cout output of each statement.

```
#include <iostream>
using namespace std;
class Number {
public:
    Number(int x, int y): real(x){
        pImaginary = new int(y);
    }
    Number(Number &cn) {
        int n= (*cn.pImaginary)+1;
        real = cn.real + 1;
        pImaginary = new int(n);
    }
    int getNumber() {
        return real+(*pImaginary);
    }
    void addImaginary(int n) {
        pImaginary += n;
    }
    void newImaginary(int n) {
        delete pImaginary;
        pImaginary = new int[n];
        setImaginary(n);
    }
    int getReal() { return real; }
    int getImaginary(int i) {
        return pImaginary[i];
    }
    void compute() {
        pImaginary = &real;
    }
    void compute(Number *cn) {
        cn = new Number(3, 4);
    }
    void compute(Number &cn) {
        cn = Number(5, 6);
    }
```

```
    Number compute(Number cn, int n) {
        cn.real += n; return cn;
    }
    void deleteImaginary() {
        delete pImaginary;
    }
private:
    void setImaginary(int n) {
        for (int i = 0; i < n; i++)
            pImaginary[i] = i*2+1;
    }
    int real;
    int *pImaginary;
};
int main() {
    Number cn1(1, 2);
    cout<<cn1.getNumber()<<endl;    //Problem 6-1
    Number cn2(3, 4);
    cn2.compute();
    cout<<cn2.getNumber()<<endl;    //Problem 6-2
    Number cn3(5, 6);
    cn3.newImaginary(3);
    cout<<cn3.getReal()<<endl;      //Problem 6-3
    Number cn4(7, 8);
    cn4.newImaginary(5);
    cout<<cn4.getImaginary(3)<<endl;//Problem 6-4
    Number cn5(9, 10);
    cn5.newImaginary(7);
    cn5.addImaginary(1);
    cout<<cn5.getNumber()<<endl;    //Problem 6-5
    Number cn6(11, 12);
    cout<<cn6.compute(cn6, 1).getNumber()<<endl;
                                    //Problem 6-6
    return 0;
}
```

Problem	Answer
6-1	
6-2	
6-3	
6-4	
6-5	
6-6	

Please copy the above answer table to your answer sheet

Problem 7 [18%, each 3%]

The following C++ program is specifically designed to implement a "Door Alter System". Please trace this program and answer problems 7-1~7-6 with the correct statements. The output of this program is: "HELP! OK! Urgent! OK!".

```
#include <iostream>
#include <string>
using namespace std;
class Alert{
public:
    Alert() { warning = 0;}
    void sendAlert(int code) {
        warning = code;
        if (code==2) cout<<"Urgent! ";
        else if (code==1) cout<<"HELP! ";
    }
    bool ___________() {    // problem 7-1
        if (warning==0) return false;
        return true;
    }
private:
    int warning;
};
class Door{
public:
    Door() { status.assign("CLOSE"); }
    string getStatus() { return status;}
    void setStatus(string s) {
        status.assign(s);
    }
private:
    string status;
};
class IMonitor{
public:
    virtual ____________; // problem 7-2
};
class Monitor:public IMonitor{
public:
    Monitor(Door *d) {   door = d;    }
    int execute() {
        string s = door->getStatus();
        if (s.compare("BROKEN")==0)
            return 2;
        else if (s.compare("OPEN")==0)
            return ______;    // problem 7-3
        else return 0;
    }
private:
    Door *door;
};
class Server {
private:
    IMonitor ___________; // problem 7-4
    Alert *alert;
public:
    Server(Alert* a) {
        alert = a;
    }
    void monitor() {
        int code = doorMonitor->execute();
        if (code>0)
            alert->_________; // problem 7-5
    }
    void setMonitor(IMonitor *dm) {
        doorMonitor = ___; //problem 7-6
    }
};
void testAlert(string msg) {
    Door *door = new Door();
    Alert *alert =new Alert();
    IMonitor *monitor = new
Monitor(door);
    Server *server = new Server(alert);
    server->setMonitor(monitor);
    door->setStatus(msg);
    server->monitor();
    if (!alert->wasAlertSend())
        cout<<"OK! ";
}
int main() {
    testAlert("OPEN");
    testAlert("CLOSE");
    testAlert("BROKEN");
    testAlert("HELP");
    return 0;
}
```

Problem	Answer
7-1	
7-2	
7-3	
7-4	
7-5	
7-6	

Please copy the above answer table to your answer sheet

Problem 8 [4%]

Complete the below class diagram for Problem 7 program.