# 國立臺北科技大學九十七學年度碩士班招生考試

## 系所組別：2310 資訊工程系碩士班甲組

## 第三節　軟體設計　試題

填 准 考 證 號 碼



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

**注意事項：**

1. 本試題共六題，配分共 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 [20%]** 每小題 2 分

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

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

註：請複製此答案表格於你的答案卷中。

```
#include <stdio.h>

void test01(int a, int b){
    printf("%d, %d\n",a||b, a&&b);                    /*   Problem 1-1    */
    printf("%d, %d\n",!a&&!b, !a||!b);                /*   Problem 1-2    */
}

void test02(int d, int count){
    int a, b, c;
    double x=2.00;
    for (a=0; a < count-1; ++a) {
        c = d--;
        for (b=a+1; b <count; ++b)
            if (c < b)    c= d--;
            x+=b/c;
    }
    printf("b=%d, c=%d\n", b, c);                     /*   Problem 1-3    */
    printf("d=%d, x=%3.2f\n", d, x);                  /*   Problem 1-4    */
}

void test03(int i, int j, int number) {
    int *p=&number;
    do{
        (*p)+=i--;
        j+=4;
    }while(j<number);
    printf("%d, j=%d\n",*p, j);                       /*   Problem 1-5, 1-9  */
}

void test04(int *p, int *q, int *r) {
    int **pp=&p, **qq=&q, **rr=&r;
    q = r;          r = p;
    *p += 2;      *q = *r + 3;
    printf("%d, %d\n", *p, *q);                       /*   Problem 1-6    */
    qq = &p;      *rr = q;
    **qq = *r + **pp;
    printf("%d, %d\n", **qq, **rr);                   /*   Problem 1-7    */
}

void test05(int a[], int p1, int *p2, int *p3) {
    a[0] = 3;      p1= 2;
    *p2 = 1;      p3 = a+1;
}

void test06(void f(int a, int b, int c), int a[]) {
    f(a[0], a[1], a[2]);
    printf("a[2]=%d, a[3]=%d\n", a[2], a[3]);         /*   Problem 1-10    */
}

int main(int argc, char *argv[]) {
    int a=1, b=2, c=3, array[]={1,2,3,4};
    test01(1,0);
    test02(a, b);
    test03(a, b, c);
    test04(&a, &b, &c);
    test05(array, array[1], &array[2], array+3);
    printf("array = %d, %d\n", array[0], array[1]);   /*   Problem 1-8    */
    test06(test03, array);
    return 0;
}
```



注意：背面尚有試題

**Problem 2 [10%]** 每小題 2 分

Given an input string of Prefix Polish Form with integer constants and +, -, * and / operator, this program will calculate the result of the Prefix form. Please complete the following codes of statement (2-1)~(2-5) in the answer table.

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

註：請複製此答案表格於你的答案卷中。

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define  STACK_BOTTOM    0
#define  OPERAND         1
#define  OPERATOR        2
#define  STACK_SIZE      100
typedef union {
     double value;
     char opera;
} store_t;
typedef struct {
     store_t store;
     int     type;
} item_t;
item_t  stack[STACK_SIZE], top;

int  is_operator(char opera){
     return opera == '+' || opera == '-' || opera == '*' || opera == '/';
}

double  compute(char opera, double operand1, double operand2){
     if(opera=='+') return (operand1 + operand2);
     else if (opera=='-') return (operand1 - operand2);
     else if (opera=='*') return (operand1 * operand2);
     ____________________________________________        /*  Problem  2-1 */
}

void  initial(void){
     ____________________________________________        /*  Problem  2-2 */
     stack[top].type = STACK_BOTTOM;
}
```

```
void  push_operand(double data){
     stack[++top].type       = OPERAND;
     stack[top].store.value = data;
}

void  push_operator(char opera){
     stack[++top].type          = OPERATOR;
     stack[top].store.opera = opera;
}

double  pop_operand(void){
     ____________________                                /*  Problem  2-3 */
}

char  pop_operator(void){
     ____________________________                        /*  Problem  2-4 */
}

int  stack_top(void){
     return  stack[top].type;
}

void evaluate(char *s) {
     double operand1, operand2;
     char    opera, *p;
     initial();
     for (p = s; *p != '\0'; p++)
          if (is_operator(*p))
               ____________________                      /*  Problem  2-5 */
          else if (isdigit(*p)) {
               operand2 = *p - '0';
               while (stack_top() == OPERAND) {
                    operand1 = pop_operand();
                    opera   = pop_operator();
                    operand2 = compute(opera, operand1, operand2);
               }
               push_operand(operand2);
          }
     printf("%.2lf, ", pop_operand());
}

int main() {
     char    s[STACK_SIZE];
     strcpy(s,"+ 5 2");      evaluate(s);
     strcpy(s,"- 5 2");      evaluate(s);
     strcpy(s,"* 5 2");      evaluate(s);
     strcpy(s,"/ 5 2");      evaluate(s);
     return 0;
}
```

The output is: 7.00, 3.00, 10.00, 2.50,

**Problem 3 [18%]** 每小題 3 分

Please review the object-oriented program written in C++ below. Record your output answers for each of the problems in the corresponding box in the following table.

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

註：請複製此答案表格於你的答案卷中。

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

class Hello {
protected:
    string nameString;
public:
    Hello(string name):nameString(name) {}
    string getName() {
        return nameString;
    }
    virtual string hello() = 0;
};

class Hi : public Hello {
public:
    Hi(string name): Hello(name) {}
    string hello() {
        return "Hi, ";
    }
};
```

```
class Aloha : public Hello {
public:
    Aloha(string name): Hello(name) {}
    string getName () {
        return "<" + nameString +">";
    }
    string hello() {
        return "Aloha, ";
    }
};

ostream& operator << (ostream& os, Hello& helloObj) {
    os << helloObj.hello() << "{" + helloObj.getName() + "}";
    return os;
}
```

```
void main () {
    Hi hiObject("Alice");

    Aloha alohaObject("Bob");
    Hello* helloPtr = &alohaObject;

    Aloha* alohaPtr = new Aloha("Christ");

    cout << hiObject.hello() << hiObject.getName() << endl;          // Problem 3-1
    cout << alohaObject.hello() << alohaObject.getName() << endl;// Problem 3-2

    cout << helloPtr->hello() << helloPtr->getName() << endl;        // Problem 3-3
    cout << alohaPtr->hello() << alohaPtr->getName() << endl;        // Problem 3-4

    cout << alohaObject << endl;                                     // Problem 3-5
    cout << *alohaPtr << endl;                                       // Problem 3-6

    delete alohaPtr;
}
```

注意：背面尚有試題

**Problem 4 [12%]** 每小題 3 分

Source code of Problem 3 has been extended as below. Please record your output answers to complete the following table. You may note "N/A" (not applicable) if any statement of the code is not meaningful.

| Problem | Answer |
|---|---|
| 4-1 | |
| 4-2 | |
| 4-3 | |
| 4-4 | |

註：請複製此答案表格於你的答案卷中。

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

class Hello {
protected:
    string nameString;
public:
    Hello(string name):nameString(name) {}
    string getName() {
        return nameString;
    }
    virtual string hello() = 0;
};

class Hi : public Hello {
public:
    Hi(string name): Hello(name) {}
    string hello() {
        return "Hi, ";
    }
};
```

```
class Aloha : public Hello {
public:
    Aloha(string name): Hello(name) {}
    string getName () {
        return "<" + nameString +">";
    }
    string hello() {
        return "Aloha, ";
    }
};

class HelloPrinter {
public:
    void print(Hello* helloObject) {
        cout << helloObject->hello() << helloObject->getName() << endl;
    }
};

template<class T>
class ObjectPrinter {
public:
    void print(T helloObj) {
        cout << helloObj.hello() << helloObj.getName() << endl;
    }
};

void main () {
    Hello& alohaRef = Aloha("David");
    HelloPrinter helloPrinter;
    helloPrinter.print(& alohaRef);          // Problem 4-1
}

void main () {
    Aloha alohaObject("Eric");
    ObjectPrinter<Hello> objectPrinter;
    objectPrinter.print(alohaObject);        // Problem 4-2
}

void main () {
    Hi& hiObject = Hi("Frank");
    ObjectPrinter<Hi> objectPrinter;
    objectPrinter.print(hiObject);           // Problem 4-3
}

void main () {
    Aloha* alohaPtr = new Aloha("Grace");
    ObjectPrinter<Aloha> objectPrinter;
    objectPrinter.print(*alohaPtr);          // Problem 4-4
    delete alohaPtr;
}
```

**Problem 5 [20%]**

This problem considers the implementation of the design pattern **Strategy**, which defines a family of algorithms, encapsulates each one, and makes them interchangeable. Suppose, as the code shown below, we have a container class, which is capable of reading, printing, and sorting a number of elements.

```
class container  {
public:
        ~container();
        void read_elements();
        void print_elements();
        void selection_sort();
private:
        int *element;
        int number_element;
};

void container::read_elements()  {
        cout << "Please input the number of elements: ";
        cin >> number_element;
        element = new int [number_element];
        for (int i = 0; i < number_element; i++)
        {
                cout << "element[" << i << "] = ";
                cin >> element[i];
        }
}

void container::print_elements()  {
        cout << "The elements are:\n" ;
        for (int i = 0; i < number_element; i++)
                cout << "element[" << i << "] = " << element[i] << endl;
}

void container::selection_sort()  {
        for (int i = 0; i < number_element-1; i++) {
                int min = element[i];
                int min_index = i;
                for (int j = i+1; j < number_element; j++) {
                        if (element[j] < min) {
                                min = element[j];
                                min_index = j;
                        }
                }
                // missing a few lines of code here
        }
}

int main()  {
        container c;
        c.read_elements();
        c.selection_sort();
        c.print_elements();
}
```

CS 2-5

(a) In the implementation of the container::selection_sort member function, there are a few lines of code that are **missing**. Please write down the missing code (please write only the code that is missing). [4%]

(b) Please give a **proper implementation** of the **destructor** of the container class. [4%]

(c) Due to the need of applying different sorting algorithms to the container, we decide to extract the container::selection_sort member function into a separate class. Following the **Strategy** pattern, we define a sorter class to encapsulate the sorting algorithm and rename our container class as the class new_container. The declarations of these classes and the new main program are as follows. Please give an implementation for the member function selection_sorter::sort. [4%]

```
class new_container  {
public:
        ~new_container();
        void read_elements();
        void print_elements();
        void sort(sorter *);
        int *element;
        int number_element;
};

class sorter    {
public:
        virtual void sort(new_container &c) = 0; // sort the elements in the container
};

class selection_sorter : public sorter  {
public:
        void sort(new_container &c);
};

int main()  {
        new_container n;
        n.read_elements();
        n.sort(new selection_sorter());
        n.print_elements();
}
```

(d) Our C++ compiler encounters a problem when it is parsing the above class declaration. The compiler reports that the parameter sorter of the new_container::sort member function is undefined. How can this problem be resolved? (Note: we can not simply put the declaration of sorter in front of new_container, because the same problem occurs.) [4%]

(e) Please give a suitable implementation for the new_container::sort member function. [4%]

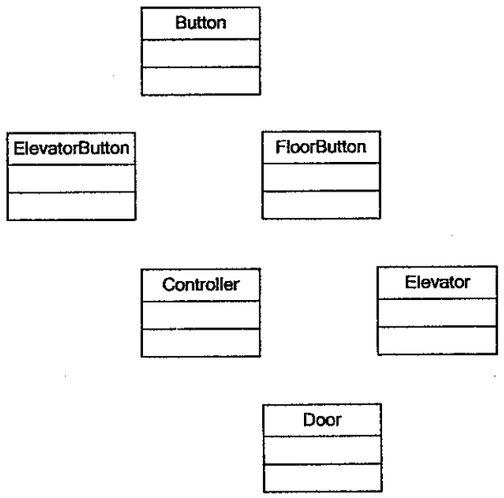
注意：背面尚有試題

**Problem 6 [20%]**

You have been appointed to develop a system that controls the elevators in the NTUT Complex Building. Basically, the system requires to move $n$ elevators between $m$ floors according to the following constraints:

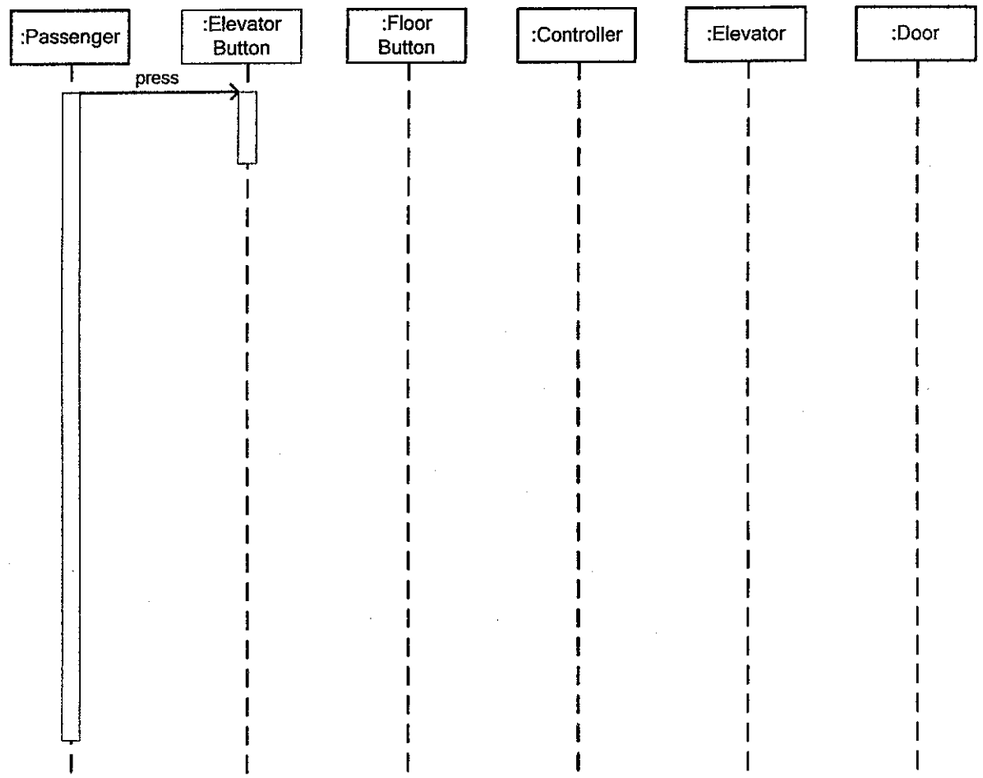
- Each elevator has a set of $m$ **floor buttons**, one for each floor. These buttons illuminate when pressed and cause the elevator to visit the corresponding floor. The illumination is canceled when the corresponding floor is visited by the elevator.
- Each floor, except the first floor and the top floor, has two **elevator buttons**, one to request an up-elevator and one to request a down-elevator. The first floor has only one elevator button to request an up-elevator and the top floor has one elevator button to request a down-elevator. These elevator buttons illuminate when pressed. The illumination is canceled when an elevator visits the floor and then moves in the desired direction.
- When an elevator has no requests, it remains at its current floor with its door closed.

(a) Six classes (Button, ElevatorButton, FloorButton, Controller, Elevator, and Door) have been identified. The Controller is responsible for the control of all buttons, elevators, and doors. Please complete the following UML class diagram for the elevator control system. Give essential **attributes** and **methods** for each class and specify their **visibility**. Show appropriate **relationships** between the classes and specify the **multiplicity** for each relationship. [10%]




註：請複製此類別圖於你的答案卷中。

(b) Complete the following UML sequence diagram for the scenario that a passenger at the first floor presses the ElevatorButton and requests to move up to the top floor with the elevator system. For simplicity, you may assume that there is only one passenger in the system and an elevator is initially at the first floor with its door closed. [10%]




註：請複製此循序圖於你的答案卷中。

CS 2-6