國立臺北科技大學 102 學年度碩士班招生考試

系所組別：2300 資訊工程系碩士班

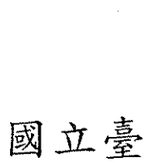
第三節　程式設計　試題

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 [12%, each 3%]

The picture below shows the output result of the following main() program in which the printUp() function prints the upper half of the picture and the printDown() function prints the lower half. Please answer problems 1-1~1-4 with the appropriate statements.



```
#include <stdio.h>
void printUp(int *x, int *y) {
    int i, j;
    for(i = 1; i <= 4; i++)
        for(j = 1; j <= 8; j++) {
            if(j==(*y)||j==((*y)+(*x)))
                printf("●");
            else
                printf("○");
        }
        if(i < 4){
            (*x)=_______;  //-----(1-1)
            (*y)=_______;  //-----(1-2)
        }
        printf("\n");
    }
}
void printDown(int x, int y) {
    int i, j;
    for(i = 1; i <= 4; i++)
        for(j = 1; j <= 8; j++)
            if(j == y || j == (y + x))
                printf("●");
            else
                printf("○");
        }
        if(i < 4) {
            x= _______;  //-----(1-3)
            y= _______;  //-----(1-4)
        }
        printf("\n");
    }
}
int main() {
    int x=1, y=4;
    printUp(&x, &y);
    printDown(x, y);
    return 0;
}
```

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

Please copy the above answer table to your answer sheet.

Problem 2 [18%, each 3%]

Given the program below in C, please trace the program and answer problems 2-1~2-6 with the printf outputs.

```
#include <stdio.h>
typedef enum
    {red, black=2, yellow, green}
color_t;
int test01(int x) {
    int r=1;
    if(x > 10) {
        r=test01(x%5)+test01(x-5);
    }
    return r;
}
int test02(int p[]) {
    int i=1;
    p[0] = 5;
    while(i<5) {
        p[i++] = p[i-1]+2*i;
    }
    return (p[i-2]);
}
int test03() {
    int i=0, j=0, k=0;
    for (i=7,j=-1;(i>0)&&(j<5);i--){
       switch (i%4) {
          case 0: k=k+(i--);
          case 1: k=k+(i--); break;
          case 2: k=k+(j++); break;
          default: k=k+(i--); break;
       }
    }
    return k;
}
int test04(int A[3][3]) {
   int i;
   A[0][2] = A[0][1];
   for(i=0; i<3; i++)
       A[i][1]=A[A[i][i]][i]+A[i][1];
   return A[1][2];
}
int test05(int q(int *), int *p) {
    return (p[0]+q(p));
}
int test06(color_t coco[]) {
   int i=0;
   for (i=0; i<4; i++)
      if (coco[i]==3) return i;
}
int main() {
    int p[5], r[]={1,2,3,4,5};
    int w[3][3] = {{0, 1, 1},
                   {1, 0, 1},{0, 1, 2}};
    char color[][7]={"red", "black",
                "yellow", "green"};
    color_t co[]={red, black, yellow, green};
    printf("%d\n",test01(20));        //-----(2-1)
    printf("%d\n",test02(p));         //-----(2-2)
    printf("%d\n",test03());          //-----(2-3)
    printf("%d\n",test04(w));         //-----(2-4)
    printf("%d\n",test05(test02,r)); //-----(2-5)
    printf("%s\n",color[test06(co)]);//-----(2-6)
    return 0;
}
```

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

Please copy the above answer table to your answer sheet.

注意：背面尚有試題

Problem 3 [24%, each 3%]

Please trace the following C programs and answer problems 3-1~3-8 with the appropriate statements.

(1) A circular Queue is implemented as below. The output is: Full! 1 2 3 Empty!

```
#include <stdio.h>
#define SIZE 4
int addQ(int data[], int front, int rear, int key){
     if ((_____)%SIZE==front) //------(3-1)
         return -999;
     data[_____] = key;       //------(3-2)
     return rear;
}
int delQ(int data[], int *front, int rear){
    if ((_____)==rear)        //------(3-3)
        return -999;
    return (data[_____]);     //------(3-4)
}
```

```
int main() {
     int front=0,rear=0,i=0,k=0;
     int Q[SIZE];
     for (i=0; i<4; i++) {
        k=addQ(Q,front,rear,i+1);
        if(k==-999) printf("Full! ");
        else rear=k;
     }
     for (i=0; i<4; i++) {
        k= delQ(Q, &front, rear);
        if(k==-999)printf("Empty! ");
        else printf("%d ",k);
     }
     return 0;
}
```

(2) A program implemented matrix transformation is given below. Assume that M1 is the output of the generate() function, and the transform() function transforms M1 into M2.

```
M1=    1 2 3 4 5          M2=    21 16 11 6 1
       6 7 8 9 10                22 17 12 7 2
       11 12 13 14 15            23 18 13 8 3
       16 17 18 19 20            24 19 14 9 4
       21 22 23 24 25            25 20 15 10 5
```

```
#include <stdio.h>
#define SIZE 5
void transform(int M1[][SIZE], int M2[][SIZE]) {
    int i=0, j=0;
    for (i=0; i<SIZE; i++)
        for (j=0; j<SIZE; j++)
            M2[____][_____]=M1[i][j]; //----(3-5)(3-6)
}
void generate(int M1[][SIZE]) {
    int i=0;
    for (i=0;i<SIZE*SIZE; i++)
        M1[_____][_____]=i+1;        //----(3-7)(3-8)
}
```

```
int main() {
    int M1[SIZE][SIZE];
    int M2[SIZE][SIZE];
    int i,j;
    generate(M1);
    transform(M1, M2);
    return 0;
}
```

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

Please copy the above answer table to your answer sheet.

Problem 4 [12%, each 2%]

Given the program below in C++, please trace the program and answer problems 4-1~4-6 with the cout outputs.

```
#include <iostream>
using namespace std;
class Being {
    protected:
    int id;
    public:
    Being(int data = 0){
        id = data;
    }
    int whatValue() {
        return getValue();
    }
    virtual int getValue() {
        return id;
    }
    void virtual printMe() {
        cout<<"B"<<endl;
    }
};
class Animal : public Being {
    public:
    Animal(int data = 1) {
        id = data;
    }
    virtual int getValue() {
        return id*2;
    }
    void virtual printMe () {
        cout<<"A"<<endl;
    }
};
```

```
class Human : public Animal {
    public:
        Human(int data=3) {
        id = data;
    }
};
int main() {
    Being *b = new Being();
    Animal *a= (Animal*) b;
    b-> printMe ();                    //--(4-1)
    delete a,b;
    b = new Animal();
    a=(Animal*)b;
    a-> printMe ();                    //--(4-2)
    Being &bref = *b;
    bref.printMe();                    //--(4-3)
    Human h(4);
    cout <<b->getValue()<<endl;        //--(4-4)
    cout <<h.Being::getValue()<<endl;  //---(4-5)
    cout <<a->Animal::whatValue()<<endl;//--(4-6)
    return 0;
}
```

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

Please copy the above answer table to your answer sheet.

Problem 5 [9%, each 3%]

Given the program below in C++. Assume that the output of the program is:

```
Mary
coffee:milk:sugar:stir
teaMilk:jelly:pearl:shake
```

Please trace the program and answer problems 5-1~5-3 with the appropriate statments.

```
#include <iostream>
using namespace std;
class Drink {
    private:
       string name, parts;
    public:
    Drink(string s) { name=s; };
    void setBrew(string b) { parts+=(":"+b);}
    void setFlavor(string f) {
        parts+=(":"+f);
    }
    void setMix(string m) {parts+=(":"+m);}
    string getMethod() {return (name+parts);}
};
class DrinkMaker {
    protected:
    Drink *drink;
    public:
    virtual void brew()=0;
    virtual void flavor()=0;
    virtual void mix()=0;
    Drink * getDrink() {return drink;}
};
class CoffeeMaker: public DrinkMaker {
    public:
    CoffeeMaker() {
      drink = new Drink("coffee");
    }
    virtual void mix() {
        drink->setMix("milk");
    }
    virtual void flavor() {
        drink->setFlavor("sugar");
    }
    virtual void brew(){
        drink->setBrew("stir");
    }
};
class TeaMilkMaker : public DrinkMaker {
    public:
    TeaMilkMaker() {
        drink = new Drink("teaMilk");
    }
    virtual void mix() {
        drink->setMix("jelly");
    }
    virtual void flavor() {
        drink->setFlavor("pearl");
    }
    virtual void brew(){
        drink->setBrew("shake");
    }
};
class Waiter {
    private:
    _____;        //--(5-1)
    string name;
    public:
    Waiter(string n) {name=n;}
    void setMaker(DrinkMaker *dMaker) {
        drinkMaker=dMaker;
    }
    void makeDrink() {
        drinkMaker->mix();
        _____;  //--(5-2)
        drinkMaker->brew();
    }
    string getName() {return name; }
    Drink *getDrink() {
       return _____;//--(5-3)
    }
};
int main() {
    DrinkMaker *teaM = new TeaMilkMaker();
    DrinkMaker *coffM = new CoffeeMaker();
    Waiter *w = new Waiter("Mary");
    cout<<w->getName()<<endl;
    w->setMaker(coffM);
    w->makeDrink();
    cout<<w->getDrink()->getMethod()<<endl;
    w->setMaker(teaM);
    w->makeDrink();
    cout<<w->getDrink()->getMethod()<<endl;
    return 0;
}
```

Problem	Answer
5-1	
5-2	
5-3	

Please copy the above answer table to your answer sheet.

Problem 6 [25%]

Consider the `Employee` class shown in Figure 1. The class has six attributes including name, job type, monthly salary, professional allowance, commission, and bonus. For simplicity, the class defines only the `Employee()`, `getMonthlySalary()`, and `getAnnualBonus()` methods where `Employee()` is a constructor method for creating an employee instance, `getMonthlySalary()` returns the values of the monthly salary attribute, and `getAnnualBonus()` gets the annual bonus of employee. Specifically, the annual bonus of an employee is computed based on the employee job type. Table 1 shows the formula to compute the annual bonus for different type of employee.

(a) [6%] Given the partial implementation of the `Employee` class in Figure 2, please use a **switch statement** to implement the method `getAnnualBonus()` in C++.

(b) [5%] To make the `Employee` class more extensible, you have redesigned the class to take advantage of polymorphism. As shown in Figure 3, an **abstract base class** `Employee` is defined. The base class has attributes name and monthly salary as well as methods `Employee()`, `getMonthlySalary()`, and `getAnnualBonus()` that are common for all the employees. Moreover, for each job type of employee, a subclass derived from `Employee` is created, such as `Engineer`, `Salesman`, or `Manager`. Each subclass has its own implementation of the constructor method and the `getAnnualBonus()` method. Based on the design in Figure 3 and the partial C++ code of the `Employee` class in Figure 4, please complete the **class definition** of `Employee` (i.e., the header file `Employee.h`).

(c) [14%] Based on the design in Figure 3, please write the C++ code of **the derived subclass** `Engineer` (i.e., both the `Engineer.h` and `Engineer.cpp`).

Employee
-_name : string -_jobType : int -_monthlySalary : int -_professionalAllowance : int -_commission : int -_bonus : int
+Employee(in name : string, in jobType : int, in monthlySalary : int, in professionalAllowance : int, in commission : int, in bonus : int) +getMonthlySalary() : int +getAnnualBonus() : int

Figure 1. The initial design of the `Employee` class

Table 1. The annual bonus formula for different type of employee

Job Type	Annual Bonus
Engineer	monthly salary + professional allowance
Salesman	monthly salary + commission
Manager	monthly salary + bonus

注意：背面尚有試題

```
// Employee.h
#include <iostream>
#include <string>
using namespace std;
using std::runtime_error;

class Employee {
private:
    string _name;
    int _jobType;
    int _monthlySalary;
    int _professionalAllowance;
    int _commission;
    int _bonus;
public:
    Employee(string, int, int, int, int, int);
    virtual ~Employee();
    int getMonthlySalary();
    int getAnnualBonus();
};

// Employee.cpp
#include "Employee.h"
const int ENGINEER = 0;
const int SALESMAN = 1;
const int MANAGER = 2;

Employee::Employee(string name, int jobType, int monthlySalary, int
professionalAllowance, int commission, int bonus) {
    _name = name;
    _jobType = jobType;
    _monthlySalary = monthlySalary;
    _professionalAllowance = professionalAllowance;
    _commission = commission;
    _bonus = bonus;
}

Employee::~Employee() {
   // empty destructor
}

int Employee::getMonthlySalary(){
    return _monthlySalary;
}

int Employee::getAnnualBonus(){
    // declare a local variable to store the annual bonus of employee
    int _annualBonus = 0;

    // compute the annual bonus using "switch statement" based on job type
     ...

    return _annualBonus;
}
```

Figure 2. The initial C++ code of the `Employee` class

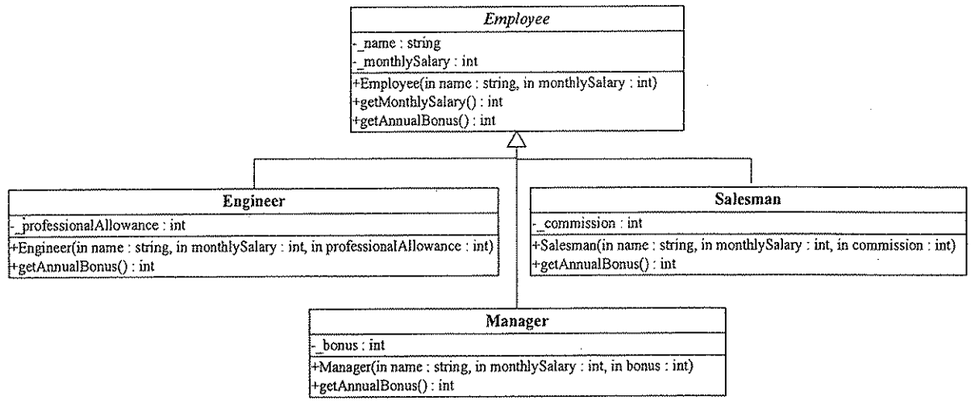


Figure 3: The refined design of the `Employee` class

```
// Employee.h
#ifndef EMPLOYEE_H_
#define EMPLOYEE_H_

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

class Employee {

    // complete the Employee code
    ...

};

#endif

// Employee.cpp
#include "Employee.h"

Employee::Employee(string name, int monthlySalary) {
    _name = name;
    _monthlySalary = monthlySalary;
}

Employee::~Employee() {
     // empty destructor
}

int Employee::getMonthlySalary(){
    return _monthlySalary;
}
```

Figure 4. The partial C++ code of the abstract base class `Employee`