國立臺北科技大學九十八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

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 [15%]

Given the program below in C. Please trace the program and fill the **1-1~1-8** blanks with the printf output of each statement.

```
#include <stdio.h>
void test01() {
     int i[5] = {2, 0, 1, 5};
     int *pt;
     pt = &i[1];
     printf("pt=%d\n", *pt);                          // Problem 1-1 [1%]
}
void test02(){
     typedef enum {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}
     weekday_t;
     weekday_t day = Monday + Saturday;
     printf("day=%d\n", day);                         // Problem 1-2 [2%]
}
void test03(int i, int j, int k) {
     do{
          i+=j--;
     }while(k<i);
     printf("i=%d, j=%d\n",i, j);                     // Problem 1-3 [2%]
}
void test04() {
     int i, z, x, y;
     for(i=1; i<5; i++)
     {
          switch(i) {
               case 1: x = 1;
                    break;
               case 2: y = 1;
                    break;
               default:
                    z = x + y;
                    y = x;
                    x = z;
                    break;
          }
     }
     printf("z=%d\n", z);                             // Problem 1-4 [2%]
}
void test05(int a, int b){
     int num;
          if (a&&b)
               num = (!a||!b) + (!a&&b);
          else
               num = (!a||b) + (a&&!b);
     printf("num=%d\n", num);                         // Problem 1-5 [2%]
}
void test06(int a[][3], int b[]){
     int* p;
     p=b;
     printf("ans=%d\n", *(p+1));                      // Problem 1-6 [2%]
     printf("a[1][0]=%d\n", a[1][0]);                 // Problem 1-7 [2%]
}
double test07(double val, int pow){
     if(pow == 0)
          return(1.0);
     else
          return(test07(val, pow - 1) * val);
}
int main(int argc, char *argv[]) {
     int a=1, b=2, c=3, array1[]={1,2,3,4}, array2[][3]={{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
     test01();
     test02();
     test03(a, b, c);
     test04();
     test05(a, b);
     test06(array2, array1);
     printf("r=%f\n", test07((double)b, c));          // Problem 1-8 [2%]
     return 0;
}
```

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

Please copy the above answer table to your answer sheet.

注意：背面尚有試題

2310

Problem 2 [15%]

Given the program below in C. Please trace the program, fill the **2-5, 2-8, 2-9** blanks with the printf output of each statement, and fill the other blanks with the correct statements.

```
#include <stdio.h>
__(2-1)__ __(2-2)__ {                                  // Problem 2-1 [1%], Problem 2-2 [1%]
char name[6];
int product[4][4];
} Employee;
void f1(Employee *x) {
     int i, *a;
     a=x->product[0];
     for (__(2-3)__; i >= 0; i--)                      // Problem 2-3 [1%]
          a[i] = __(2-4)__;                            // Problem 2-4 [2%]
     printf("%d,%d,%d\n",a[0],a[2],a[3]);   /* Print out 1, 5, 7 */
}
void f2(char *x, char *y, int n) {
     int i;
     for (i=0; i<n; i++, x++, y++)
          if (*x!=*y) *x = *y+1;
     *x='\0';
     printf("%s \n", x-3);                             // Problem 2-5 [2%]
}
void f3(int t[][4]) { /* matrix multiplication */
     int k, c, r;
     int m[][3] ={{1,1,1},{2,2,2},{1,1,2}};
     int col[][3]={{1,2,3},{2,3,4},{3,4,5}};
     for (r=0; r<3; r++) {
          for (c=0; c<3; c++) {
               t[c][r]=0;
               for (k=0; k<3; k++)
                    t[c][r] += __(2-6)__*__(2-7)__; // Problem 2-6[2%], Problem 2-7[2%]
          }
     }
     printf("%d,%d,%d \n",t[0][0],t[0][1], t[0][2]);  /* Print out 6,9,12 */
     printf("%d,%d \n",t[1][0],t[1][1]);               // Problem 2-8 [2%]
}
int main() {
     Employee e1;
     f1(&e1);
     f2(e1.name, "abcd", 4);
     printf("->%s\n", e1.name);                         // Problem 2-9 [2%]
     f3(e1.product);     return 0;
}
```

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

Please copy the above answer table to your answer sheet.

Problem 3 [20%, each 1%]

Please trace the following C++ program and provide the result of the cout statement.

(a) Using objects

```
class X {
public:
     X(int v=5) { value = v; }
     X(X &src)  { value = src.value * 3; } // copy constructor
     void compute1(int a)      { value *= a; }
     void compute2(int a)      { value /= a; }
     void compute3(X x)        { value += x.value; }
     X compute4()              { X x;return x; }
     void calculate(int a)     { value -= a; }
     void calculate(double a) { value = (int) (value + a); }
     void print()              { cout << value << " "; }
private:
     int value;
};

void Problem3a()
{
     X x1; x1.print();                                  // Problem 3a-1
     x1.compute1(2);x1.print();                         // Problem 3a-2

     X x2(2);x2.print();                                // Problem 3a-3
     x2.compute2(4);x2.print();                         // Problem 3a-4

     X x3(3); X x4; x4 = x3;x4.print();                 // Problem 3a-5
     X x5 = x3; x5.print();                             // Problem 3a-6

     X x6(6); x6.calculate(5); x6.print();              // Problem 3a-7
     X x7(7); x7.calculate(4.5);x7.print();             // Problem 3a-8

     X x8(8), x9(9);x8.compute3(x9); x8.print();        // Problem 3a-9
     x8.compute4().print();                             // Problem 3a-10
}
```

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

Please copy the above answer table to your answer sheet.

(b) Object as parameters

```
class P
{
public:
     P(int v1, int v2)  {value1 = v1; value2 = new int(v2); }
     void Add(int x)    { value1 += x; *value2 += x; }
     P operator+(P rhs) {
          value1 += rhs.value1 + 1; *value2 += *rhs.value2 + 2; return *this; }
     void print1()      {cout << value1 << " "; }
     void print2()      {cout << *value2 << " ";}
     int value1, *value2;
};

class Q
{
public:
     void Compute1(P p) { p.Add(2); }
     void Compute1(P *p){ p->Add(3); }
     void Compute2(P *p1, P *p2){ P *t = p1; p1 = p2; p2 = t; }
     void Compute2(P &p1, P p2) { p1 = p2; }
     void Compute3(P *p1, P &p2, P p3) { *p1 = p2 + p3; }
};

void Problem3b()
{
     Q q;
     P p1(1,2); q.Compute1(p1); p1.print1();              // Problem 3b-1
     p1.print2();                                         // Problem 3b-2

     P p2(2,3); q.Compute1(&p2);p2.print1();              // Problem 3b-3
     p2.print2();                                         // Problem 3b-4

     P p3(3,4), p4(4,5);q.Compute2(&p3, &p4); p3.print1(); // Problem 3b-5

     P p5(5,6), p6(6,7);q.Compute2(p5, p6);p5.print2();   // Problem 3b-6
     p6.Add(10); p5.print2();                             // Problem 3b-7

     P p7(7,8), p8(8,9), p9(9,10);
     q.Compute3(&p7, p8, p9); p7.print1();                // Problem 3b-8
     p7.print2();                                         // Problem 3b-9
     p8.print2();                                         // Problem 3b-10
}
```

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

Please copy the above answer table to your answer sheet.

A5-2-2

Problem 4 [20%, each 2%]

Please trace the following C++ program and provide the result of the cout statement.

```
class Base
{
public:
     Base(int b1=0, int b2=0) { value1 = b1; value2 = b2; }
     void compute(int x) { value1 += x; value2 *= x; }
     void compute1(int x) { calculate(x); }
     virtual void calculate(int x) {value1 = x;value2 = x * x; }
     void print() { cout << value1+value2 << " "; }
     int value1, value2;
};

class Derived : public Base
{
public:
     Derived(int x1, int x2, int x3) : Base(x1, x2)
          { value2 = x1+x2; value3 = x3; }
     virtual void calculate(int x) { value1 = x;value2 = 2 * x;value3 = x+1; }
     void print() { cout << value2+value3 << " "; }
     int value2, value3;
};

void Problem4()
{
     Base b1(1,2); b1.print();                            // Problem 4-1
     b1.compute(3); b1.print();                           // Problem 4-2

     Derived d1(2,3,4); d1.print();                       // Problem 4-3
     d1.compute(5); d1.Base::print();                     // Problem 4-4

     Derived d2(3,4,5); Base b2;b2 = d2; b2.print();      // Problem 4-5

     Derived d3(4,5,6); d3.calculate(3); d3.print();      // Problem 4-6
     d3.Base::print();                                    // Problem 4-7

     Derived d4(5,6,7); Base *b4; b4=&d4; b4->compute(2); b4->print();// Problem 4-8
     b4->calculate(4); d4.print();                        // Problem 4-9

     Base *b5 = new Derived(6,7,8); b5->compute1(5); b5->print();  // Problem 4-10
}
```

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

Please copy the above answer table to your answer sheet.

注意：背面尚有試題

Problem 5 [15%]

You are asked to use the **Observer** pattern to implement a system that handles two account balances. The partial program and two sample test results of the system are given below. Let the abstract class **Subject** provide the interface for adding and removing observers to the concrete Subject (**Account**). The view of account balance data is separated from the Account class (**Subject**) and is implemented in the concrete Observer class (**AccountTextView**). When the account balance is changed by the class **AccountInput**, the view will be updated automatically. Please complete the following programs for designing the system with the observer pattern.

(a) Implement void Account::setBalance(double) [5%]

(b) Implement void AccountTextView::update() [5%]

(c) Implement void AccountManager::createAccountView(Account, Account) [5%]

Two test results of the program are given as below.

TestCase1: deposit 200 into Account 1, and withdraw 200 from Account 2	TestCase2: withdraw 3000 into Account 1, and withdraw 2000 from Account 2
(0) Cancel. (1) Deposit. (2) Withdraw. Account 1-choose (0/1/2):1 amount:200 -Account 1, Balance->1200 -Account 2, Balance->3000 (0) Cancel. (1) Deposit. (2) Withdraw. Account 2-choose (0/1/2):2 amount:200 -Account 1, Balance->1200 -Account 2, Balance->2800	(0) Cancel. (1) Deposit. (2) Withdraw. Account 1-choose (0/1/2):2 amount:3000 (0) Cancel. (1) Deposit. (2) Withdraw. Account 2-choose (0/1/2):2 amount:2000 -Account 1, Balance->1000 -Account 2, Balance->1000

The part of the program is shown as follows:

```
class Subject;
class Observer{
     public:
     virtual void update()=0;
};
class Subject {
     private:
     list<Observer*> observers;
     public:
     void addObserver (Observer* o) {
          observers.push_back(o); }
     void delObserver (Observer* o) {
          observers.remove(o);    }
     void notify() {
          list<Observer*>::iterator pos = observers.begin();
          while (pos != observers.end())  {
               ((Observer* )(*pos))->update();
               ++pos;
          }
     }
};
class Account: public Subject {
     private:
     string name;      // Account name
     double balance;  // Account balance
     void setBalance(double);     //problem 5(a)
     public:
     Account(string name, double opening ) {
          this->name = name;
          setBalance(opening);
     }
     double getBalance() { return balance; }
     void withdraw(double m) {
          if ((m>0)&&(getBalance()-m> = 0))
               setBalance( getBalance()-m);
     }
     void deposit( double m) {
          if (m>0) setBalance(getBalance()+m);
     }
     string getName() { return name; }
};
class AccountInput {
     private:
     Account *account;
     public:
     void setAccount(Account *account) {
          this->account = account;
     }
     void input(){
          int choose;     double amount;
          cout<<"(0) Cancel. (1) Deposit. (2) Withdraw."<<endl;
          cout<<account->getName()<<"-choose (0/1/2):";  cin >> choose;
          if ((choose!=1) && (choose!=2)) return;
          cout << "amount:";     cin >> amount;
          if (choose==1) account->deposit(amount);
          else account->withdraw(amount);
     }
};
class AccountTextView: public Observer {
     private:
     Account* account1, *account2;
     public:
     AccountTextView(Account* a1, Account* a2) {
          this->account1 = a1;
          this->account2 = a2;
     }
     void update();          // problem 5(b)
};
class AccountManager {
     public:
     AccountManager() {
          Account account1("Account 1", 1000.0);
          Account account2("Account 2", 3000.0);
          createAccountView(account1, account2);
     }
     // setup AccountTextView, AccountInput
     // and invoke AccountTextView::input() to accept deposit or withdraw
     void createAccountView(Account, Account);     //problem 5(c)
};
int main () {
     AccountManager accountManager;   return 0;
}
```

Problem 6 [15%]

Consider a simple transaction-processing system for a retailer that sells a wide array of computer parts, such as memory and motherboard. A partial UML class diagram of the system is given below, where the **Customer** class can invoke the *getTotal()* method of the **Sale** class to obtain the total cost of the parts being sold in each transaction. For simplicity, the implementation of *getTotal()* keeps every part being sold in an array ***cart*** and iteratively adds up the total cost of the parts as shown in the class diagram. The **Part** class is defined as an abstract class and the **Memory** and **Motherboard** are its concrete subclasses. This design has a problem. During the annual sales or promotions, the retailer may have special offers and discounts on different parts. As a result, the retailer has to modify the *getTotal()* method, which can be cumbersome and error-prone. To ease this problem, a better design is to define a discount policy class. Each part then can have discount policies to set the price discount of the part, and hence, the *getTotal()* method can be closed for modification.

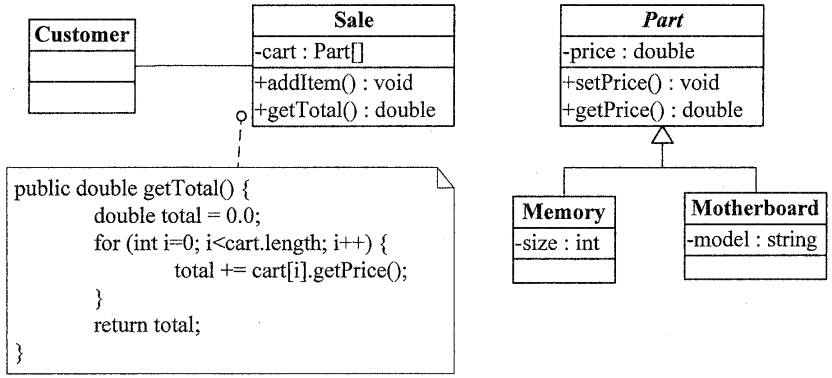

Figure: The partial UML class diagram of the transaction-processing system

(a) Assume that an abstract **DiscountPolicy** class is defined with an attribute *discount* which is a double number representing the discount factor of the price. Each part can have a concrete discount policy. Please extend the partial UML class diagram for the transaction-processing system with two types of discount policies: **AnnualSales** and **SpecialPromotion**. 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%]

(b) Based on your design in (a), please implement the **Part** class in C++. [5%]