國立臺北科技大學九十九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. 本試題共四題，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 [30%, each 3%]

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.

```
#include <stdio.h>
int f1(int a) { return ++a; }
int f2(int a, int b) {
    if (a==0 || b==0) return 1;
    else if (a<b) a = f2(a, b-1);
    else a = f2(a-1, b);
    return a;
}
int f3(int a, int b) {
    if (a>b) return (a+b);
    else return (a-b);
}
int f4(int x[], int a) {
    int i=0, result=0;
    for (i=0; i<=a; i++) result = x[i]/2;
    return result;
}
void test01() {
    int index=1, key=9;
    do {
        index++;   key--;
    } while (index<=6) ;
    printf("%d\n", index++);            // Problem 1-1
    printf("%d\n", key);                // Problem 1-2
}
void test02() {
    int n=5, m=4;
    printf("%d\n", f1(n++));            // Problem 1-3
    printf("%d\n", f2(n, m));           // Problem 1-4
    printf("%d\n", f3(m, n));           // Problem 1-5
}
void test03() {
    int key=0, y[6], x[]={5, 4, 3, 2, 1, 0}, i=0, j=0;
    while (x[i])  {
        if ((i++)< 3)  y[j++]= x[i];
        else y[j++]=-2;
    }
    key = f4(x, x[0]);
    printf("%d\n", y[0]);               // Problem 1-6
    printf("%d\n", y[2]);               // Problem 1-7
    printf("%d\n", key);                // Problem 1-8
}
void test04() {
    char temp, s[]={'a', 'b', 'c', 'd', 'e', 'f', 'g'};
    int i, j;
    for (i=1; i<=5; i++)
        for (j=2; j<=5; j++)
            if (s[j-1]>=s[j]) {
               temp = s[j-1];  s[j-1]=s[j];  s[j]=temp;
            }
    printf("%c\n", s[j-2]);             // Problem 1-9
    printf("%c\n", s[i-3]);             // Problem 1-10
}
int main() {
    test01();
    test02();
    test03();
    test04();
}
```

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

Please copy the above answer table to your answer sheet.

注意：背面尚有試題

Problem 2 [22%, each 2%]

Please trace the following C++ program and provide the results of the cout statements.

```
class Vehicle
{
public:
    Vehicle(int speed, string name = ""): _name(name + "Vehicle")
        { _speed = speed; }
    string Name() { return _name;}
    double Speed() { return _speed; }
    void Accelerate(double x) {
        _speed += x;
        if (_speed < 0)
           _speed = 0;
    }
    virtual void SlowDown() { _speed = 8; }
    virtual int Wheels() { return 2; }
protected:
    double _speed;
    string _name;
};

class Car : public Vehicle
{
public:
    Car(string name = "Car::"): Vehicle(10, name) {}
    void Accelerate(double x) {
        Vehicle::Accelerate(x);
        if (x > 200)
           _speed = 200;
    }
    virtual void SlowDown() { _speed = 20; }
    virtual int Wheels() { return 4;}
};
```

```
void Problem2()
{
    Vehicle v(5);
    cout << v.Name() << endl;              // Problem 2-1
    cout << v.Wheels() << endl;            // Problem 2-2
    v.Accelerate(10);
    cout << v.Speed() << endl;             // Problem 2-3

    Car c;
    cout << c.Name() << endl;              // Problem 2-4
    c.Accelerate(215);
    cout << c.Speed() << endl;             // Problem 2-5
    c.Vehicle::SlowDown();
    cout << c.Speed() << endl;             // Problem 2-6

    Vehicle *v_ptr = new Car("RaceCar");
    cout << v_ptr->Name() << endl;         // Problem 2-7
    cout << v_ptr->Wheels() << endl;       // Problem 2-8
    v_ptr->SlowDown();
    cout << v_ptr->Speed() << endl;        // Problem 2-9
    v_ptr->Accelerate(-25);
    cout << v_ptr->Speed() << endl;        // Problem 2-10
    v_ptr->Accelerate(250);
    cout << v_ptr->Speed() << endl;        // Problem 2-11
}
```

Problem	Answer	Problem	Answer
2-1		2-2	
2-3		2-4	
2-5		2-6	
2-7		2-8	
2-9		2-10	
2-11			

Please copy the above answer table to your answer sheet.

Problem 3 [28%]

An integer number is said to be a ***prime*** number if it is divisible by only 1 and itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not. The number 1 is by definition not a prime number.

(a) [7%] To test if an integer n is a prime number, one approach is to loop through 2 to n-1 and test whether each number divides exactly into n. If any of them do, the number n is not a prime number. Based on this approach, write a C/C++ function `isPrime()` that can test if an integer is a prime number. The function `isPrime()` will take an input parameter `num` of type integer and return 0 if `num` is not a prime number; otherwise return 1.

(b) [7%] Write a C/C++ program `main` that uses the function `isPrime()` to find the first 1,000 prime numbers. The `main` program also has to store these prime numbers with an integer array `primeArray` of size 1,000 and calculate their sum.

(c) [7%] A more efficient approach to test if an integer n is a prime number is to test the divisions of only the prime numbers less than or equal to $\sqrt{n}$ (instead of looping from 2 to n-1). If n is divisible by any prime number less than or equal to $\sqrt{n}$, then n is not a prime number. For example, to determine if 25 is a prime number, we only need to check if 25 is divisible by 2, 3, and 5. Calculate the sum of the first 1,000 prime numbers with this approach by rewriting the `main` program and `isPrime()` function in (b) such that `isPrime()` takes three input parameters: `num`, `primeArray`, and `sizeofprimeArray`, where `num` is the number to be tested, `primeArray` is an integer array storing the prime numbers less than `num`, and `sizeofprimeArray` is the size of `primeArray`.

(d) [7%] If n is large and all the prime numbers less than or equal to n are to be found, using the approach given in (c) may be slow. A faster approach is to avoid using division operations, and use only addition operations. The idea is to loop through 2 to n-1 and, in iteration i, mark all multiples of i as non-prime. At the end, the numbers that are not marked non-prime are prime numbers. An example is given below. A `PrimeFlag` array of type `bool` is defined, and every element of the array is initialized as `true` (T). A loop from 2 to n-1 is conducted. When the loop count is i, the program reports that i is a prime number, if `PrimeFlag[i]` is `true`, and marks the `PrimeFlag` of $2i$, $3i$, $4i$, ... as `false` (F); the program does nothing, if `PrimeFlag[i]` is `fasle`. For example, when the loop count is 2, 2 is reported prime and the `PrimeFlags` of 2, 4, 6, 8, ... are marked as `false`; when the loop count is 3, 3 is reported prime and the `PrimeFlags` of 3, 6, 9, 12, ... are marked as `false`; when the loop count is 4, nothing is done because the `PrimeFlag[4]` is `false`; ... By using this approach, please write a C/C++ program `main` that reads an input number n from console and prints all prime numbers $\leq n$ to the console. Note that your program **should not contain any multiplication or division operations**.

Initialization

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
PrimeFlag	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	...

Loop count 2: report 2 is prime (2's `PrimeFlag` is `true`) and mark 4, 6, 8, ...as non-prime

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
PrimeFlag	T	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	...

Loop count 3: report 3 is prime (3's `PrimeFlag` is `true`) and mark 6, 9, 12, ... as non-prime

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
PrimeFlag	T	T	F	T	F	T	F	F	F	T	F	T	F	T	F	T	F	T	F	...

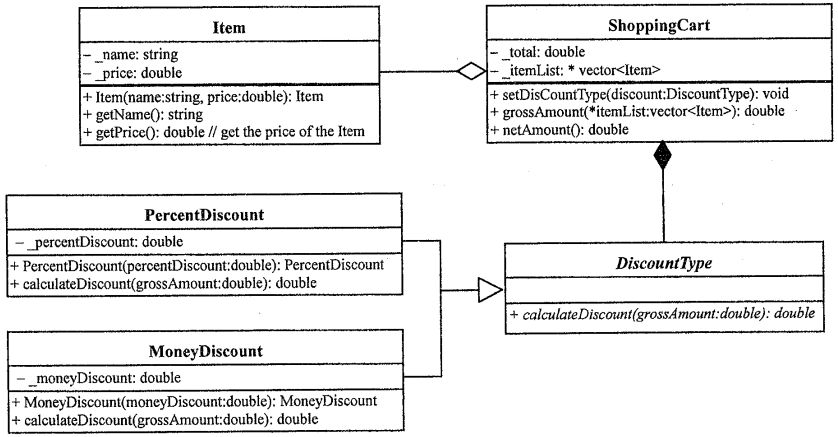
Loop count 4: Do nothing because 4's `PrimeFlag` is False

Loop count 5: report 5 is prime (5's `PrimeFlag` is `true`) and mark 5, 10, 15, ... as non-prime

...

Problem 4 [20%]

You are asked to implement a shopping cart for a wholesale store. The design of the shopping cart is given below.



Basically, the shopping cart takes a list of items and calculates the gross amount by adding up the price of each item (implemented in `ShoppingCart.grossAmount()`). It then applies discount to the gross amount depending on the types of discounts that customers have (implemented in `ShoppingCart.netAmount()`). The wholesale store offers different types of discounts, such as percentage discount (`PercentDiscount`) and fixed dollar amount discount (`MoneyDiscount`). To avoid many if-else statements in `ShoppingCart.netAmount()`, an abstract `DiscountType` class with an abstract method `calculateDiscount()` is created. The `calculateDiscount()` method will take the gross amount and apply discount so as to calculate the net amount of the purchase. Different types of discounts will be the subclasses of `DiscountType` and implement their own `calculateDiscount()` method in order to deduct proper discount off the gross amount. For example, for $100 gross amount, the net amount should be $80 if applying 20% percentage discount and should be $90 if applying $10 money discount.

注意：背面尚有試題

(a) [6%] Given the partial implementation of the shopping cart below, please complete the implementation of method ShoppingCart. grossAmount (). in C++.

(b) [14%] Please complete both header (.h) and body (.cpp) for DiscountType, MoneyDiscount, and PercentDiscount classes in C++. Note that, in class MoneyDiscount, if the gross amount is less than the value of _moneyDiscount (i.e., fixed dollar discount), calculateDiscount() method will simply return 0.

```
// ***** main.cpp
#include <iostream>
#include <string>
#include <vector>
#include "ShoppingCart.h"
#include "Item.h"
#include "MoneyDiscount.h"
#include "PercentDiscount.h"
using namespace std;

int main(int argc, char* argv[]) {
   vector<Item> itemList;
   Item item1("food", 100);  // create item 1(name= "food" , price=$100)
   itemList.push_back(item1);
   Item item2("drink", 150); // create item 2(name= "drink" , price=$150)
   itemList.push_back(item2);

   ShoppingCart shoppingCart; // create shopping cart & calculate the gross amount
   cout << "gross amount : " << shoppingCart.grossAmount(&itemList) << "\n";

  //create and apply $20 money discount
   MoneyDiscount discount1(20.0);
   shoppingCart.setDiscountType(&discount1);
   cout << "net amount after money discount: " << shoppingCart.netAmount() << "\n";

   //create and apply 20% percentage discount
   PercentDiscount discount2(0.8);
   shoppingCart.setDiscountType(&discount2);
   cout << "net amount after percent discount: " << shoppingCart.netAmount();

   return 0;
}

// ***** ShoppingCart.h
#include "DiscountType.h"
#include "Item.h"
#include <vector>

class ShoppingCart {
private:
    double _total;
    vector<Item> *_itemList;  //vector to store a list of Items
    DiscountType * _discount;
public:
    ShoppingCart();
    virtual ~ShoppingCart();
    void setDiscountType(DiscountType *);
    double grossAmount(vector<Item> *);
    double netAmount();
};
// ***** ShoppingCart.cpp
#include "ShoppingCart.h"
void ShoppingCart::setDiscountType(DiscountType *discount) {
    _discount = discount;
}

double ShoppingCart::grossAmount(vector<Item> *itemList) {
   // ititialize _itemList
   ...

   // declare an iterator used to traverse the _itemList vector
   ...

   _total=0; // initialize the gross amount
   // use a for loop and the iterator to get each item from the _itemList
   // get the price of each item and add to the gross amount _total
      ...
   return _total;
}

double ShoppingCart::netAmount() {
   return _discount->calculateDiscount(_total);
}

//***** DiscountType.h
      ...

//***** DiscountType.cpp
      ...

//***** MoneyDiscount.h
#include "DiscountType.h"
class MoneyDiscount: public DiscountType {
      ...
};

//***** MoneyDiscount.cpp
#include "MoneyDiscount.h"
MoneyDiscount::MoneyDiscount(double moneyDiscount) {
    _moneyDiscount = moneyDiscount;
}
// MoneyDiscount::calculatetDiscount
      ...

//***** PercentDiscount.h
#include "DiscountType.h"
class PercentDiscount: public DiscountType {
      ...
};

//***** PercentDiscount.cpp
#include "PercentDiscount.h"
PercentDiscount::PercentDiscount(double percentDiscount) {
    _percentDiscount = percentDiscount;
}
// PercentDiscount::calculateDiscount
      ...
```