國立臺北科技大學九十五學年度碩士班招生考試

系所組別：1810 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. 請按順序標明題號作答，不必抄題。
3. 全部答案均須答在答案卷之答案欄內，否則不予計分。

Problem 1 [20%]

Given the program below in C. (1) Please tract the lines 1~3 and answer the questions 1-1 ~ 1-3. (2) Please tract the lines 1~17 and fill the 1-4 ~ 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	

Note: please copy this table to your answer sheet.

本題共計 10 小題，配置 20 分，每小題 2 分。

1-1 Which statement is correct? (1) p=s; (2) p =&s; (3) p=s[0]; (4) p = *s;

1-2 If all of the elements of array s have been properly initialized, which statement can correctly retrieve the value of a specific element? (1) s[5] (2) s (3) **s (4) *(s+1)

1-3 Which statement is correct? (1) p = malloc(x*size(char)); (2) p = (char *) malloc(x*size(char)); (3) p = (char *) malloc(y*sizeof(char)); (4) z = (int *) malloc(x*sizeof(int));

```
1. char s[4];
2. char *p;
3. int x=5, y=0, *z, *q, j=6;
4. int num[2][3]={{1,2,3}, {4,5,6}};
5. z = &x;
6. q = z;
7. printf("%d \n", *q);                    // problem 1-4
8. printf("%x \n", &s[0]);                 // output: 240ff5c
9. printf("%x \n", &s[1]);                 // output: 240ff5d
10. printf("%x \n", &s[2]);                // problem 1-5
11. printf("%x \n", &s[3]);                // problem 1-6
12. z = &num[0][0];
13. printf("%d \n", *(z+2));               // problem 1-7
14. ++z;
15. printf("%d \n", *(z+2));               // problem 1-8
16. printf("%d \n", *z);                   // problem 1-9
17. printf("%d \n", *++z);                 // problem 1-10
```

注意：背面尚有試題

Problem 2 [15%]

Given the program below in C, please identify 5 incorrect codes (except line 0 and 1). You should give the line number of the error codes and correct them (e.g. 0. #include <stdio.h>). The C program implements a Stack. If the input is 4, 1, 0, 2, 0, 0, 0, the output must be =>1, =>2, =>4, stack underflow.

本題共計 5 小題，配置 15 分，每小題 3 分。

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

Note: please copy this table to your answer sheet.

```
0.   include stdio.h;
1.   int *p, *top;
2.   void push(int i) {
3.         p++;
4.         if (p= (top+50)) {
5.               printf("stack overflow");
6.               exit(1);
7.         }
8.         *p=i;
9.   }
10.  int pop( ) {
11.       if (p==top) {
12.             printf("stack underflow\n");
13.             exit(1);
14.       }
15.       p--;
16.       return   (p+1);
17.  }
18.  int main (void) {
19.       int value;
20.       p= (int *) malloc (50*sizeof(int));
21.       if (p) {
22.             printf("allocation failure !\n");
23.             exit(1);
24.       }
25.       top = p;
26.       do {
27.             scanf("%d", value);
```

```
28.             if (value!=0) push(value);
29.             else printf("=> %c \n", pop( ));
30.         } while (value!=-1);
31.         return 0;
32.     }
```

Problem 3 [20%]

Given the object-oriented program below in C++, please tract the code and fill the following blanks with the output of each main function. You may note "N/A" (not applicable) if the statement is not meaningful.

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

註 1：請複製此答案表格於你的答案卷中。
註 2：本題共計 10 小題，配置 20 分，每小題 2 分。
註 3：每小題答對者得 2 分，對錯者倒扣 1 分；未作答者得 0 分。

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

class Animal{

protected:
  int position;

public:
  Animal(int pos) {
      position = pos;
  }

  void walk() {
      position ++;
  }

  virtual void run() = 0;

  void location() {
      cout << position << endl;
  }

};
```

```
class Bear : public Animal{

public:
  Bear(int pos):Animal(pos) { }

  void walk(){
      position += 2;
  }

  void run(){
      position *= 2;
  }
};


class Tiger : public Animal{

public:
  Tiger(int pos):Animal(pos) { }

  void walk(){
      position += 10;
  }

  void run(){
      position *= 10;
  }
};
```

```
void main(){
   Animal *pAnimal = new Animal(10);
            pAnimal->walk();
                  pAnimal->location();      // problem 3-1 (2%)
            pAnimal->run();
                  pAnimal->location();      // problem 3-2 (2%)
   delete pAnimal;
}
```

```
void main(){
   Animal *pAnimal = new Bear(10);
            pAnimal->walk();
                  pAnimal->location();      // problem 3-3 (2%)
            pAnimal->run();
                  pAnimal->location();      // problem 3-4 (2%)
   delete pAnimal;
}
```

```
void main(){
   Animal *pAnimal = new Tiger(10);
            pAnimal->walk();
                  pAnimal->location();      // problem 3-5 (2%)
            pAnimal->run();
                  pAnimal->location();      // problem 3-6 (2%)
   delete pAnimal;
}
```

```
void main(){
   Bear *pBear = new Bear(10);
            pBear->walk();
                  pBear->location();        // problem 3-7 (2%)
            pBear->run();
                  pBear->location();        // problem 3-8 (2%)
   delete pBear;
}
```

```
void main(){
   Bear *pBear = new Tiger(10);
            pBear->walk();
                  pBear->location();        // problem 3-9 (2%)
            pBear->run();
                  pBear->location();        // problem 3-10 (2%)

   delete pBear;
}
```

注意：背面尚有試題

Problem 4 [15%]

A C++ program is designed in the next page. Please tract the code and give the output of statement (4-1) ~ (4-15) in the following answer table.

Problem	Answer			
4-1	**ID =**		**Value =**	
4-2	**ID =**		**Value =**	
4-3	**ID =**		**Value =**	
4-4	**ID =**		**Value =**	
4-5	**ID =**		**Value =**	
4-6	**ID =**		**Value =**	
4-7	**ID =**		**Value =**	
4-8	**ID =**		**Value =**	
4-9	**ID =**		**Value =**	
4-10	**ID =**		**Value =**	
4-11	**ID =**		**Value =**	
4-12	**ID =**		**Value =**	
4-13	**ID =**		**Value =**	
4-14	**ID =**		**Value =**	
4-15	**ID =**		**Value =**	

註 1：請複製此答案表格於你的答案卷中。

註 2：本題共計 15 小題，配置 15 分，每小題 1 分。

註 3：各小題 ID 及 Value 答案全答對者得 1 分，答錯任一者得 0 分。

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

class Data{

private:
      static int counter;
      int id;
      int value;

public:
      Data() {
            id = ++counter;
      }

      Data(int a):value(a) {
            id = ++counter;
      }

      Data(const Data& data) {
            id = ++counter;
            value = data.value;
      }

      Data& operator = (const Data& data) {
            value = data.value;
            return *this;
      }

      Data operator + (const Data& data) {
            int sum = value + data.value;
            return Data(sum);
      }

      void setValue1(Data data) {
            value = ++data.value;
      }

      void setValue2(Data& data) {
            value = ++data.value;
      }

      void setValue3(Data* data) {
            value = ++data->value;
      }

      void printValue() {
            cout << "ID = " << id <<
                  "    Value = " << value << endl;
      }
};
```

```
int Data::counter=0;

void main(){

      Data d1(5);
      Data d2 = d1;
      Data& d3 = d2;
      Data d4;

      d4 = d3;
            d1.printValue();    // problem 4-1 (1%)
            d2.printValue();    // problem 4-2 (1%)
            d3.printValue();    // problem 4-3 (1%)
            d4.printValue();    // problem 4-4 (1%)

      d4.setValue1(d1);
            d1.printValue();    // problem 4-5 (1%)
            d4.printValue();    // problem 4-6 (1%)

      d4.setValue2(d2);
            d2.printValue();    // problem 4-7 (1%)
            d4.printValue();    // problem 4-8 (1%)

      d4.setValue3(&d3);
            d3.printValue();    // problem 4-9 (1%)
            d4.printValue();    // problem 4-10 (1%)

      Data d5 = d1 + d2;
            d2.printValue();    // problem 4-11 (1%)
            d5.printValue();    // problem 4-12 (1%)

      d5 = d3 + d4;
            d3.printValue();    // problem 4-13 (1%)
            d4.printValue();    // problem 4-14 (1%)
            d5.printValue();    // problem 4-15 (1%)
}
```

Problem 5 [20%]

An **Iterator** design pattern is used when we want to access the elements of an aggregate object sequentially without exposing its underlying implementation. The following C++ code fragment is an implementation of a simple List object:

```
class List {
friend class ListIterator; // allow ListIterator to access private members
private:
        int *list_data;       // pointer to an array storing the list data
        int list_size;        // size (number of elements) of the list data
public:
        List(int size) {      // size defines the number of elements of the list
                list_size = size;
                list_data = new int [size];
        }
        ~List() {
                delete [] list_data;
        }
};
```

Now we want to use the interface Iterator defined below to access the elements of a List object. An example usage is shown in the main() program below, which assigns the contents of a list as 0, 1, 4, … and then prints out the list.

```
class Iterator {
public:
        virtual void First()=0;     // Initialize the current element as the first element
        virtual void Next()=0;      // Advance the current element to the next element
        virtual bool IsDone()=0; // Return true if the current element is beyond
                                 //              the last element
        virtual int &CurrentItem()=0;// Return the reference of the current element
};

void main()
{
        int i; List list(10);                          // create a 10-element list
        Iterator *iter = new ListIterator(&list); // iter is an iterator to the list
        // assign the contents of the list as 0 1 4 9 16 25 36 49 64 81
        for (i = 0, iter->First(); !iter->IsDone(); iter->Next(), i++)
                iter->CurrentItem() = i*i;
        // print out the contents of the list
        for (iter->First(); !iter->IsDone(); iter->Next())
                cout << iter->CurrentItem() << " ";
        cout << endl;
}
```

Suppose the ListIterator class is defined below, please give a full implementation of the ListIterator class (including its constructor and 4 virtual functions).

```
class ListIterator : public Iterator {
public:
        …
private:
        List *list_ptr;       // pointer to the List associated with this ListIterator
        int *element_ptr; // pointer to the current element of the List
};
```

Problem 6 [10%]

Answer the following questions related to private and protected constructors in C++.

(a) What happens if the **constructor** of a class X is declared as **protected** (also called protected constructors)? What are the restrictions of using the class X? [3%]

(b) What happens if the **constructor** of a class X is declared as **private** (also called private constructors)? What are the restrictions of using the class X? [3%]

(c) When should a private or protected constructor be used? Please describe a scenario that a private or protected constructor is useful. [4%]