

114年公務人員特種考試警察人員、一般警察人員、
國家安全局國家安全情報人員、移民行政人員考試及
114年特種考試退除役軍人轉任公務人員考試試題

考試別：一般警察人員考試

等別：三等考試

類科組別：警察資訊管理人員

科目：物件導向程式設計

考試時間：2 小時

座號：_____

※注意：(一)禁止使用電子計算器。

(二)不必抄題，作答時請將試題題號及答案依照順序寫在試卷上，於本試題上作答者，不予計分。

(三)本科目除專門名詞或數理公式外，應使用本國文字作答。

一、針對下列 C++ 程式碼，請說明其輸出與其程式 p.print()、q.split()(Line 28, 29)執行之運作流程邏輯；並請說明在 testPattern()中測試 Pattern 類別 split()(Line 27, 29)，對於「指令與條件判斷涵蓋度」不足之處。(25 分)

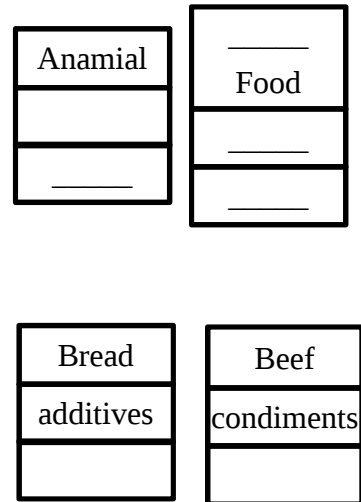
```
01 #include <iostream>
02 #include <string>
03 #include <vector>
04 using namespace std;
05 class Pattern {
06 public:
07     Pattern(string data) { this->data = data; }
08     void split(string symbol) {
09         int start = 0, stop = 0;
10         stop = data.find(symbol);
11         while (stop != string::npos) {
12             if (stop - start != 0) words.push_back(data.substr(start, stop - start));
13             start = stop + symbol.size();
14             stop = data.find(symbol, start);
15         }
16         if (start != length) words.push_back(data.substr(start));
17     }
18     void print() { for (auto& s : words) cout << s << "#";   cout<<endl; }
19 private:
20     string data;
21     vector <string> words;
22     int length;
23 };
24 void testPattern() {
25     Pattern p("this is a test");
26     Pattern q("that was so happple");
27     p.split(" ");
28     p.print();
29     q.split("ha");
30     q.print();
31 }
32 int main() {
33     testPattern();
34     return 0;
35 }
```

二、下列 Java 程式碼計算食物 (Food) 料理 (cook) 後，動物 (Animal) 擷取之卡路里量。請完成統一塑模語言 UML 圖；並說明其執行後的輸出，以及多型 (Polymorphism)、封裝 (Encapsulation) 發生所在之程式碼行數。(25 分)

```

01 abstract class Food {
02     public Food(int cal, String type) {
03         this.calorie = cal;
04         this.type = type;
05     }
06     abstract public void cook(int c);
07     public String show(int c) {
08         String result = type+":";
09         cook(c);
10         result += calorie;
11         return result;
12     }
13     protected int calorie;           // 卡路里
14     private String type;
15 }
16 class Bread extends Food {
17     public Bread(int c, int a) {
18         super(c, "Bread");
19         additives = a;
20     }
21     public void cook(int cal) { calorie += additives*cal; }
22     private int additives;
23 }
24 class Beef extends Food {
25     public Beef(int c) {
26         super(c, "Beef");
27         condiments = c*2;
28     }
29     public void cook(int cal) { calorie += condiments%cal; }
30     private int condiments;
31 }
32 class Anamial {
33     public Anamial(Food f) { food = f; }
34     public String getCal() { return food.show(2); }
35     private Food food;
36 }
37 public class FoodTest {
38     public static void main(String[] args) {
39         Food f1 = new Bread(3, 2);
40         Food f2 = new Bread(2, 3);
41         Food f3 = new Beef(3);
42         Anamial a1 = new Anamial(f1);
43         Anamial a2 = new Anamial(f2);
44         Anamial a3 = new Anamial(f3);
45         System.out.println(a1.getCal());
46         System.out.println(a2.getCal());
47         System.out.println(a3.getCal());
48     }
49 }

```



三、針對下列 C++ 程式碼，請修正程式碼行數 58 「Job jobs ...」的錯誤及修正錯誤後正確執行之輸出；並請說明 Employee 類別 candidate 資料的運作原理，以及程式碼行數 61 「e1.arrange()」程式執行之運作流程。(25 分)

```
01 #include <iostream>
02 #include <string>
03 #include <map>
04 using namespace std;
05 class Job {
06 public:
07     Job(string _name, int _sa, int _sk) {
08         name = _name;
09         salary = _sa;
10         skill = _sk;
11         employee_name = "None";
12     }
13     string getName() { return name; }
14     int getSalary() { return salary; }
15     int getSkill() { return skill; }
16     void hire(string _name) { employee_name = _name; }
17     void print() { cout << name << "::" << employee_name << endl; }
18 private:
19     string name;
20     int salary;
21     int skill;
22     string employee_name;
23 };
24 class Employee {
25 public:
26     Employee(string _name, int _sa, int _sk) {
27         name = _name;
28         salary = _sa;
29         skill = _sk;
30     }
31     void match(Job *m) {
32         if ((m->getSalary() >= salary) && (skill >= m->getSkill())) {
33             candidate[m->getName()] = m;
34         }
35     }
36     void arrange() {
37         int score = 0;
38         Job *jb, *jb_wanted;
39         for (auto c = candidate.begin(); c != candidate.end(); ++c) {
40             jb = c->second; // get value of MAP
41             if (jb->getSalary() >= score) {
42                 score = jb->getSalary();
43                 jb_wanted = jb;
44             }
45         }
46         jb_wanted->hire(name);
47     }
48     string getName() { return name; }
49 private:
50     string name;
51     int salary;
52     int skill;
53     map <string, Job *> candidate;
54 };
55 int main() {
56     Employee e1("Tom", 37, 85);
57     Employee e2("John", 35, 75);
58     Job jobs[] = { new Job("RD", 45, 75), new Job("Sales", 35, 70),
59                   new Job("Manager", 55, 80) };
60     for (int i=0; i<3; i++) e1.match(jobs[i]);
```

```

61     e1.arrange();
62     for (int i=0; i<3; i++) e2.match(jobs[i]);
63     e2.arrange();
64     for (int i=0; i<3; i++) jobs[i]->print();
65     return 0;
66 }

```

四、針對下列 C++ 程式碼，請說明其輸出與其 `inpuScore()`、`computeAverage()` 執行之運作流程邏輯；並請說明 `computeAverage()` 設計上的問題。(25 分)

```

01 #include <stdexcept>
02 #include <iostream>
03 #include <string>
04 using namespace std;
05 class Student{
06 public:
07     Student(string n) { name = n; math_score = eng_score = average = 0; }
08     void checkSmall(int s) {
09         if (s<0) throw out_of_range("small");
10         cout<<"ok,";
11     };
12     void inpuScore(int m_s, int e_s) {
13         try {
14             checkSmall(m_s); checkSmall(e_s);
15             math_score = m_s;
16             eng_score = e_s;
17         }
18         catch(exception &e) { cout<<e.what()<<endl; }
19     }
20     void computeAverage(int n) {
21         try {
22             if (n<=0) throw out_of_range("zero");
23             average = (math_score+eng_score)/n;
24         }
25         catch(exception &e) { cout<<"exc"<<endl; }
26         catch(out_of_range &e) { cout<<e.what()<<endl; }
27     }
28     void print() { cout << name <<": "<<average<<endl; }
29 private:
30     int math_score; // 數學成績
31     int eng_score;  // 英文成績
32     int average;    // 平均成績
33     string name;
34 };
35 void test01() {
36     Student stu("John");
37     stu.inpuScore(0, 0);
38     stu.inpuScore(100, -1);
39     stu.inpuScore(100, 90);
40     stu.print();
41 }
42 void test02() {
43     Student stu("Tom");
44     stu.inpuScore(100, 90);
45     stu.computeAverage(0);
46     stu.print();
47 }
48 int main() {
49     test01();
50     test02();
51     return 0;
52 }

```