1. (6 marks) Java program is given:

```
class Person{
        int x;
        public Object work(){return new Person();}
}

class Human{
        int x;
        public Object work(){ return new Person();}
}

class Worker extends Person {
        int x;
        public Object work() { return new Worker();}
        public void overTime(int h) { x = x+h; }

        public static void main(String[] args){
                Person a = new Human();         // line 1
                Worker b = new Person();        // line 2
                Person c = new Worker();        // line 3
                c.overTime(5);                  // line 4
                c.x =5;                         // line 5
                Worker m = c.work();            // line 6
        }
}
```

For each line (line 1 to line 6), does it compile? If it does not compile, give the reason.

line 1 ไม่compile เพราะผิด name Equivalent

line 2 ไม่compile เพราะ Person เป็น worker ไม่ได้ ถ้าเรียกแล้ว เพราะไม่ได้ใช้ method ที่เรียกของ Person

line3 ได้ เพราะ worker เป็นลูกของ person

line 4 ไม่ได้ เพราะ c.เป็น person ไม่สามารถเรียก function ของ worker ได้ แต่ต้อง cast ก็ได้

line 5 ได้

line 6 ไม่ได้ เพราะ return Object ออกมา

1. (6 marks) Java program is given:

return type

```
class Person{
        int x;
        public Object work(){return new Person();}
}

class Human{
        int x;
        public Object work(){ return new Person();}
}

class Worker extends Person {
        int x;
        public Object work() { return new Worker();}
        public void overTime(int h) { x = x+h; }

        public static void main(String[] args){
                Person a = new Human();         // line 1
                Worker b = new Person();        // line 2
                Person c = new Worker();        // line 3
                c.overTime(5);                  // line 4
                c.x =5;                         // line 5
                Worker m = c.work();            // line 6
        }
}
```

สร้าง

บอกประเภทของตัวแปร

For each line (line 1 to line 6), does it compile? If it does not compile, give the reason.

โครงสร้างเหมือนกัน และ ชื่อเหมือนกัน

line1 : not complie : type "Person" and "Human" not name Equivalent

line2 : not complie : "Person" (ตัวแม่) does not have all methods of "Worker" (ตัวลูก

line3 : compile

line4 : not complie : c as a type "Person" does not have method overtime.

line5 : compile

line6 : not complie : c.work( ) จะคืนค่าเป็น Object ซึ่ง Object ไม่สามารถเก็บใน m ได้
เพราะประเภทของ m (ซึ่งก็คือ Worker) มีรายละเอียดมากกว่า

เอาใหญ่ เก็บใน เล็ก ไม่ได้!

~~worker = Object~~

แม่ x = new ลูก() ✓

2. For the code below (a language with nested subroutine), the language uses a value model of variables.

```
program A(){
    x, y, z: integer;
    procedure B(){
        y: integer;
        y=0;
        x=z+1;
        z=y+2;
    }

    procedure C(){
        z: integer;
        procedure D(){
            x: integer;
            x = z + 1;
            y = x + 1;
            call B();
        }
        z = 5;
        call D();
    }
    x = 10;
    y = 11;
    z = 12;
    call C();
    print x, y, z;
}
```

A()

x = 10

y = ~~11~~ → 7

z = 12

C()

z = ~~5~~ → 2

D()

x = ~~6~~ → 5+1=6

B()

y = 0

Static

A( )

x = ~~10~~ → 13

y = ~~11~~ → 6+1 = 7

z = ~~12~~ → 0+2 = 2

C()

z = 5

D()

x = 6

~~B()~~ ← 1 ← Dynamic

B() ← 2

y = 0

Static

2.1 (3 marks) If the language uses static scoping, the printed result of x, y, and z is	2.2 (3 marks) If the language uses dynamic scoping, the printed result of x, y, and z is
x = 13	x = 10
y = 7	y = 7
z = 2	z = 12

2. For the code below (a language with nested subroutine), the language uses a value model of variables.

```
program A(){
    x, y, z: integer;
    procedure B(){
        y: integer;
        y=0;
        x=z+1;
        z=y+2;
    }

    procedure C(){
        z: integer;
        procedure D(){
            x: integer;
            x = z + 1;
            y = x + 1;
            call B();
        }
        z = 5;
        call D();
    }
    x = 10;
    y = 11;
    z = 12;
    call C();
    print x, y, z;
}
```

แบบ static scope

program A()

x = ~~10~~ ↗ 12+1 = 13

y = ~~11~~ ↗ 6+1 = 7

z = ~~12~~ ↗ 0+2 = 2

program C()

z = 5

program D()

x = 5+1 = 6

program B() ← ระวัง! ย่อหน้า ยึดตามโค้ดเลย (ถ้าเป็น static)

y = 0

แบบ dynamic scope

program A()

x = 10

y = ~~11~~ ↗ 6+1 = 7

z = 12

program C()

z = ~~5~~ ↗ 0+2 = 2

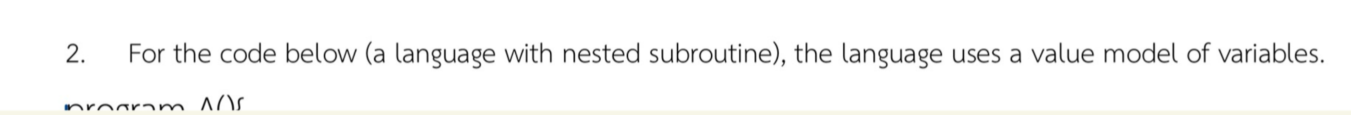
program D()

x = 5+1 = ~~6~~ ↗ 5+1 = 6

program B()

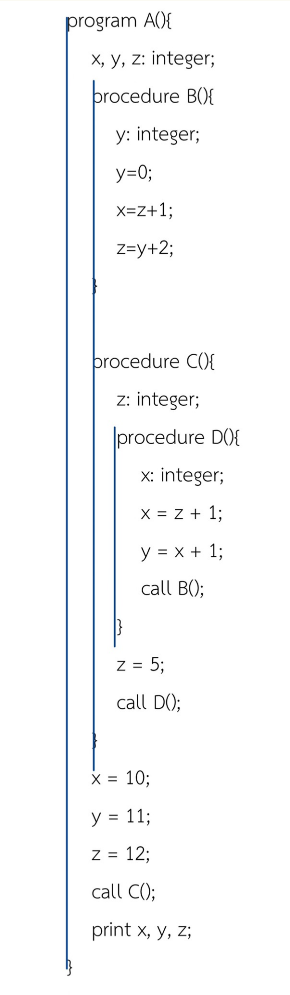
y = 0

2.1 (3 marks) If the language uses static scoping, the printed result of x, y, and z is	2.2 (3 marks) If the language uses dynamic scoping, the printed result of x, y, and z is
x = 13	x = 10
y = 7	y = 7
z = 2	z = 12



2. For the code below (a language with nested subroutine), the language uses a value model of variables.

program A(){



```
program A(){
    x, y, z: integer;
    procedure B(){
        y: integer;
        y=0;
        x=z+1;
        z=y+2;
    }

    procedure C(){
        z: integer;
        procedure D(){
            x: integer;
            x = z + 1;
            y = x + 1;
            call B();
        }
        z = 5;
        call D();
    }
    x = 10;
    y = 11;
    z = 12;
    call C();
    print x, y, z;
}
```

~~x~~ ~~y~~ ~~z~~

13 7 2

C

z: 5

D

x: 6

B

y: 0

x: 13

x ~~y~~ z

10 7 12

C

z: ~~5~~ 2

D

x: ~~6~~ 5+1

y: 7

B

new y: 0

	Static S	Dynamic S
x	13	
y	7	
z	2	

3. Given the C++ code below.

```
class First {
public:
    First() { b = 10; }
    virtual void display(int &x, int y) { x = x + y; cout << "b, x " << b << " " << x << endl; }
private:
    int b;
};

class Second: public First {
public:
    Second() { d = 20; }
    virtual void display(int &x, int y) { x = x * y; cout << "d, x " << d << " " << x << endl; }
private:
    int d;
};

int main() {
    First f, *p;
    Second s;
    int m = 1;
    int *n = new int(2);
    float o = 5.7;
    p = &s;
    p->display(m, o);          //line1
    f = s;
    f.display(m,o);            //line2
    return 0;
}
```

Method ที่ overwrite ได้

Java ทั้ง pointer เลย เวลาประกาศ (ตัวแปร pointer)

pointer เหลือที่เป็น Dynamic) ของ cpp

Object ธรรมดาเลย เป็น Static

(f เป็น Object)

(1 mark) At line1, the method binding is ☐ static ☑ dynamic

(1 mark) At line2, the method binding is ☐ static ☑ dynamic

* (1 mark) In the checking of the types of the method arguments at line2, the following rule(s) of the type system are used (you may choose 1 or more).

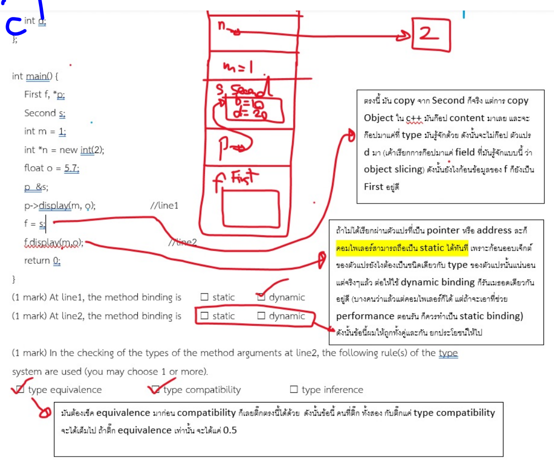
☑ type equivalence ☑ type compatibility ☐ type inference

ถูกเช็คก่อน type compatibility เสมอ

หรือ

= ใช้แทนกันได้โดยไม่ต้องมีการ Cast

การรวม type เช่น int + float ได้อะไร!



```
    int d;
};

int main() {
    First f, *p;
    Second s;
    int m = 1;
    int *n = new int(2);
    float o = 5.7;
    p_&s;
    p->display(m, o);          //line1
    f = s;
    f.display(m,o);            //line2
    return 0;
}
```

ตรงนี้ มัน copy จาก Second ก็จริง แต่การ copy Object ใน c++ มันก็อป content มาเลย และจะก็อปมาแค่ที่ type มันรู้จักด้วย ดังนั้นจะไม่ก็อป ตัวแปร d มา (เค้าเรียกการก็อปมาแค่ field ที่มันรู้จักแบบนี้ ว่า object slicing) ดังนั้นยังไงก้อนข้อมูลของ f ก็ยังเป็น First อยู่ดี

ถ้าไม่ได้เรียกผ่านตัวแปรที่เป็น pointer หรือ address ละก็ คอมไพเลอร์สามารถถือเป็น static ได้ทันที เพราะก้อนออบเจ็กต์ของตัวแปรยังไงต้องเป็นชนิดเดียวกับ type ของตัวแปรนั้นแน่นอน แต่จริงๆแล้ว ต่อให้ใช้ dynamic binding ก็รันเมธอดเดียวกันอยู่ดี (บางคนว่าแล้วแต่คอมไพเลอร์ก็ได้ แต่ถ้าจะเอาที่ช่วย performance ตอนรัน ก็ควรทำเป็น static binding) ดังนั้นข้อนี้ผมให้ถูกทั้งคู่และกัน ยกประโยชน์ให้ไป

(1 mark) At line1, the method binding is ☐ static ☑ dynamic

(1 mark) At line2, the method binding is ☐ static ☐ dynamic

(1 mark) In the checking of the types of the method arguments at line2, the following rule(s) of the type system are used (you may choose 1 or more).

☑ type equivalence ☑ type compatibility ☐ type inference

มันต้องเช็ค equivalence มาก่อน compatibility ก็เลยติ๊กตรงนี้ได้ด้วย ดังนั้นข้อนี้ คนที่ติ๊ก ทั้งสอง กับติ๊กแค่ type compatibility จะได้เต็มไป ถ้าติ๊ก equivalence เท่านั้น จะได้แค่ 0.5

4. A Java-like language uses left-to-right evalution order. Its precedence and associativity rules are given below. (Precedence is from the highest downto the lowest.)

Operator	Description	Associativity
...	...	...
* / %	multiplicative	left to right
...	...	...
== !=	equality	left to right
...	...	...
&&	logical and	left to right
||	logical or	left to right
...	...	...

4.1 (3.5 marks) Add parentheses to the expression below to show the effect of precedence and associativity to the grouping of operands to operators.

((c % 400) == 0) || (((c % 4) == 0) && ((c % 100) != 0))

false false true

4.2 (1.5 marks) If c is 1666, the result of the expression is ....false.........

4.3 (3 marks) If this language has short circuiting, which of these subexpressions get evaluated in the question 4.2?

c % 400 == 0 ☑ yes □ no

c % 4 == 0 ☑ yes □ no

c % 100 != 0 □ yes ☑ no

ไม่ได้รับการคิด เพราะพอมันเจอ false && .... มันจะ false ทันที