# SWE3006 Prog. Lang. & Compilers [2020F Final Exam]

**Name :** **Student ID#:**

**1. Write what the following paragraphs describe in programming language. [total 15 points]**

**(A)** With one symbolic name, multiple different function implementations exist in the program. Depending on context, one of the functions are used. [5 points]

**(B)** When boolean expressions are evaluated, logically connected terms are partially evaluated. As the compiler can determine the final outcomes with partial evaluation, the execution speed can be improved. [5 points]

**(C)** For a given expression, the compiler determines the type of the overall expression with their components' types in the expression. This process is needed for the next step, where compatible type related operations are added properly. [5 points]

**2. Considering the following pseudo code for scope and binding [total 25 points]**

```
x: integer := 1       --- global
y: integer := 2

procedure add
      x := x + y
procedure second(P : procedure)
      x: integer := 2
      P()
procedure first
      y: integer := 3
      second(add)

-- body of main program
first()
write_integer(x)   -- print out the value of x, which is passed as call-by-value
```

(NOTE) When passing a procedure as an argument, two types of binding exist

- * shallow binding – use the referencing environment where the procedure is executed. This mode is default for most languages.
- * deep binding – current referencing environment is bundled together with a reference to procedure and pass this as an argument. Then, use the bundled referencing environment, when the procedure is executed.

**(A)** What does the code print if language uses static scoping? [5 points]

**(B)** What does the code print if language uses dynamic scoping with *shallow binding*? [10 points]

**(C)** What does the code print if language uses dynamic scoping with *deep binding*? [10 points]

**3. Considering to the following class declarations in Java-like language, where we assume all methods are virtual and all members are public by default, answers the questions.** [**total 45 points**] (NOTE) assume the sizes of pointer and int are 4 byte.

```
class Polygon {
  private:
    int sides, symmetric;
  public:
    int IsSymmetric();
};
class Rectangle {
  private:
    int sideLength1, sideLength2;
  public:
    int Area();
    int NumSides();
};
Class Square extends Polygon, Rectangle {
    Int Area();
};
```

**(A)** When multiple inheritance is allowed as shown in the code, describe relations with a diagram among class instance (with layout), method table, and method implementation for the three classes. (15 points)

**(B)** For three class instances, we may assign its reference (y) to other typed reference variable (x) in a program. Some cases are allowed by the compiler and the other cases are not allowed. In the following table, mark cases with O, if allowed and X, if not allowed.

In addition, among the allowed cases, simply copying address is not enough to make type casting work correctly. Indicate such cases and specify extra operations needed. (10 points)

x = (*type*) y;

| x y | Polygon | Rectangle | Square |
|---|---|---|---|
| Polygon | | | |
| Rectangle | | | |
| Square | | | |

**(C)** When implementing methods, some methods require *trampoline functions*, explain why you need them with an example. (10 points)

(D) Draw a diagram for method tables and method implementation to use trampoline functions, as answered in (C). You also write a pseudo code for the body of trampoline functions. (10 points)

**4. Considering the following program with garbage collection, specify data/heap/stack segments separately, then show variables in root set, heap objects, and their pointer links [total 15 points]**

```
class Node {
    int val;
    Node next;
    Node(int i) {val = i; next = null;};
};
public class List {
    static Node head = null;
    static public void main(String args[]) {
        Node temp;
        int i;
        for (i=0; i<5; i++) {
            insert(new Node(i));
        }
        /* ----- (i) ----------------- */
        temp = remove(1);
        /* ----- (ii) ---------------- */
        remove(4);
        /* ----- (iii) --------------- */
    }; /* -- end of main --*/
```

```
    static void insert(Node n) {
         n.next = head;
         head = n;
    };
    static Node remove (int i) {
        Node cur = head;
        Node prev = null;
        while (cur != null) {
            if (cur.val == i) {
                if (prev == null) head = cur.next;
                else    prev.next = cur.next;
                break;
            }
            prev = cur;
            cur = cur.next;
        }
        return cur;
   };
}; /* -- End of Class List --*/
```

**(A)** When main() proceeds to the point (iii), show variables, allocated objects in each segment, and pointer links among them. Based on sweep phase of GC, mark live objects. [10 points]

**(B)** When main() returned but the program is not terminated, repeat the drawing as in (A) and mark live objects. [5 points]