Section I: Multiple choice questions (10 points in total, each question is weighted 1 point)

1. What is the name of the category of programming languages whose structure is dictated by the von Neumann computer architecture?

A. Imperative　B. Denotational
C. Functional　D. Non-procedural
E. Constraint　F. Object-oriented

2. A paradigm that allows specification of what has to be computed rather than just how a computation is to be carried out.

A. Imperative　B. Denotational
C. Functional　D. Non-procedural
E. Constraint　F. Object-oriented

3. A paradigm incorporating encapsulation, inheritance, and dynamic type binding.

A. Imperative　B. Denotational
C. Functional　D. Non-procedural
E. Constraint　F. Object-oriented

4. Atttribute grammars are used to specify

A. basic syntax of a programming language
B. non-finite state machines
C. static semantics of a programming language
D. dynamic semantics of a programming language

5. Which language is considered to be the first fully object-oriented language?

A. Fortran　B. COBOL
C. LISP　D. C
E. JAVA　F. SMALLTALK

6. In what language is UNIX written?

A. FORTRAN　B. COBOL
C. LISP　D. C
E. JAVA　F. SMALLTALK

7. What programming language has dominated scientific computing over the past 35 years?

A. FORTRAN　B. COBOL
C. LISP　D. C
E. JAVA　F. SMALLTALK

8. What language has the slogan "Write once, run anywhere"?

A. FORTRAN　B. COBOL
C. LISP　D. C
E. JAVA　F. SMALLTALK

9. What programming language has dominated artificial intelligence programming over the past 35 years?

A. FORTRAN　B. COBOL
C. LISP　D. C
E. JAVA　F. SMALLTALK

10. What programming language has dominated business applications over the past 35 years?

A. FORTRAN　B. COBOL
C. LISP　D. C
E. JAVA　F. SMALLTALK

Section II: Matching (20 points in total, each question is weighted 1 point)
At the left, write the number of the term that best fits the definition. Illegible or ambiguous entries will be marked incorrect.

1. ____ a method used to create a new object 2. ____ a linked list in which the rear item refers back to the head item 3. ____ an association between a variable attribute and an entity. 4. ____ the meaning of an expression 5. ____ holds the information on the call stack needed for one procedure call 6. ____ a mechanism for hiding and protecting information 7. ____ a mechanism for hiding detail 8. ____ an area of memory from which space for dynamic structures is allocated 9. ____ a data structure of values linked together in memory by a chain of references 10. ____ a data structure with last-in first-out behavior, supporting push and pop 11. ____ the act of changing the values in variables or data structures 12. ____ encapsulates data and knows how to operate on the data 13. ____ a recasting of one problem as another (somewhat different) problem 14. ____ how to carry out a computation, may be implemented as a procedure 15. ____ a data structure with first-in first-out behavior 16. ____ a property of the loop variables that is true initially and after each iteration 17. ____ a predicate that becomes true when a computation ends 18. ____ a property of the internal representation true before and after each method executes 19. ____ the notation used to express an idea 20. ____ memory that had been allocated but is no longer reachable	(1). abstraction (2). activation record (3). algorithm (4). binding (5). circular list (6). constructor (7). encapsulation (8). garbage (9). heap (10). linked list (11). loop invariant (12). mutation (13). object (14.) queue (15.) reduction (16). representation invariant (17). semantics (18). stack (19). syntax (20). termination condition

Section III: Programming examples (30 points in total, each question is weighted 3 points)

1. In C and C++ the concepts of "expression" and "command" are often confused. For instance, `--x` is an expression, in the sense that it returns a result, and also a command, in the sense that it changes the state of x. Expression whose evaluation might affect the state are called "expressions with side effects", and require special care by the programmer as they cannot be regarded as "pure values". Consider for instance the following (incorrect) definition of the factorial function:

```
int fact(int x) {
   if (x == 0) return 1;
   else return  fact(--x) * x;
}
```

(1) What is the result returned by the call `fact(3)` ?
(2) Would the definition be correct if we substitute the command `return fact(--x) * x;` with `{ int y = x; return fact(--x) * y; }` ?

2. Consider the following procedure declaration in C:

```
void p() {int x; q = &x; *q = 1; }
```

where `q` is a global variable declared as pointer to `int` and initialized by `q = new int`. What output can we expect to be produced by the following piece of code:

```
*q = 0 ; p() ; printf(*q) ;
```

3. Consider the following sequence of declarations:

```
z : integer;
procedure p (x:integer) x := x+1 ; z := z+2;
```

The following piece of code

```
z := 1 ; p(z) ; write(z)
```

Wat are the final outcome of variable z in case of **pass by value, pass by reference, and pass by value-result.**

4. Consider the following procedure declaration:

```
procedure p;
    begin
      x: integer;
      procedure q; x := x+1;
      procedure r;
        begin
          x: integer in x := 1; q ; write(x)
        end
    in
      x:= 2; r
    end
```

What is the output produced by calling p in a language with:
(1) static scope
(2) dynamic scope

5. Consider the following class declarations in C++:

```
class C {
  private: int x;
  protected: int y;
  public: int z;
};

class C1: public C {
  private: int x1;
  public: int y1;
};

class C2: private C {
  private: int x2;
  public: int y2;
};
```

Assume that in `main` there is the following declaration:

```
C1 o1;
```

For each of the following expressions in `main`, say whether they are allowed or not (they can be forbidden either because they violate the class definition or the protection mechanism)

expression in main	correct or incorrect	if incorrect, explanation
o1.x		
o1.y		
o1.z		
o1.x1		
o1.y1		
o1.x2		
o1.y2		

本試題雙面印製

6. Provide a concise description of foo's functionality.

```
boolean foo(Rectangle r, Point p) {
   return ( (p.x == r.x) &&
            (p.y >= r.y) &&
            (p.y < r.y + r.height) );
}
```

7. Provide a concise description of baz's functionality.

```
double baz(double[] x, double[] y) {
   int n = Math.min(x.length, y.length);
   int i = 0;
   double s = 0;
   while (i < n) {
     s = s + x[i] * y[i];
     i++;
   }
   return s;
}
```

8. Assuming that the class Stack is defined to encapsulate a stack of integers with the usual semantics, what are the values of the variables v, w, x, y, and z after execution of the following code?

```
Stack s = new Stack();
s.push(5);
s.push(6);
s.push(7);
int v = s.pop();
s.push(8);
s.push(9);
int w = s.peek();
int x = s.pop();
int y = s.pop();
int z = s.pop();
```

variable	value
v	
w	
x	
y	
z	

The following two procedures are syntactically correct but have logic errors. For each one, identify the error, explain what would happen as a result of the error, and show how to fix the error.

9. The following procedure is supposed to compute the sum of all the positive odd numbers up to n.

```
int sumOdd(int n) {
   if ((n % 2) == 0)   // make n odd
      n--;
   if (n == 0)   // check for base case
      return 0;
    else  // otherwise recurse and combine
      return (n + sumOdd(n-2));
   }
```

10. The following procedure is supposed to compute the sum of all the integers in an array.

```
int sum(int[] a) {
   int k = 0;
   int s = 0;
   while (k < a.length) {
      s += a[k];
      k++;
   }
   return k;
}
```

Section IV: Programming examples (40 points in total, each question is weighted 10 points)

Problem 1: What is the output of the following Java program? The program compiles and runs correctly as shown.

```
public class C1 {
   public C1() {
      System.out.println("Hello from C1");
   }
   public int v1() {return( 3 );}
   public int v2() {return( 1 );}
   static public void main(String[] args) {
   System.out.println("Hello from main");
       C3 x = new C3();
       C2 y = new C2();
       C1 z = new C1();
       System.out.println ( x.v1() );
       System.out.println ( y.v2() );
       System.out.println ( z.v1() );
       System.out.println ( x.v2() );
       System.out.println ( y.v1() );
       System.out.println ( z.v2() );
   }
}
class C2 extends C1 {
   public int v1() {return( 2 );}
   public int v2() {return( 4 );}
   public C2() {
      System.out.println("Hello from C2");
   }
}

class C3 extends C2 {
   public int v2() {return( 6 );}
   public C3() {
      System.out.println("Hello from C3");
   }
}
```

Problem 2: What is the output of the following Java program? The program compiles and runs correctly as shown.

```
class Teacher {
   private String name;    // Teacher's name
   private int    size;    // Size of class
   public Teacher (String theName) {
      size = 0; name = theName;
   }
   public Teacher (int theSize, String theName)
     {this.setSize(theSize); name = theName;}
   public int getSize () {return size;}
   public int setSize (int n) { return size = n;}
   public void print(){
      System.out.println(name+" has "
                         +size+" students.");
   }
   public void compare (Teacher theTeacher) {
      System.out.println (name + " says:");
      if (theTeacher.size>16)
 System.out.println("  You have many students.");
      if (size < 17)
 System.out.println("     I   don't   have   many
students.");
      if (theTeacher.size<size)
  System.out.println ("  I have more than you.");
      else if (theTeacher.size == size)
   System.out.println ("  We have the same.");
   else
    System.out.println("  I have fewer than you.");
    }
    public void enroll () {size++;}
    public void enroll (int n) {
      size = size+n;
    }
}

public class School {
   public static void main(String args[]) {
      Teacher one = new Teacher(18, "ABC");
      one.print();
      Teacher two = new Teacher("XYZ");
      two.print();
      two.enroll(16);
      two.print();
      one.compare (two);
      two.compare (one);
      one.enroll (); one.print ();
      one.enroll (3); one.print ();
      two = one;
      one.enroll();
      two.enroll();
      one.print ();
      two.print ();
   }
}
```

本試題雙面印製

Problem 3: Expressions

Consider the following program in C syntax:

```
int foo (int *i) {
  *i *= 3;
  return 1;
}

void main () {
  int x = 5;
  x = x + foo (&x);
  if ((x++) || (foo (&x)))
    x+=10;
  printf("%d\n",x);
}
```

What is the output of this program with the following individual assumptions:

(1). Left-to-right operand evaluation, short-circuited boolean expressions

(2). Left-to-right operand evaluation, non-short-circuited boolean expressions

(3). Right-to-left operand evaluation, short-circuited boolean expressions

(4). Right-to-left operand evaluation, non-short-circuited boolean expressions

Problem 4: Scoping

Consider the following code fragment in a new language currently under design called ALIBABA, which has Pascal-like syntax except that all procedures have no formal parameters. The designers cannot decide whether to adopt **dynamic** or **static scoping**.

```
program test;
    var a, b: integer;
    procedure sub1;
      var a, c, e: integer;
    begin
      a := 6; c := 7; e := 8;
      ...              {position three}
    end; {sub1}
    procedure sub2;
      var a, c, d: integer;
    begin
      a := 3; c := 4;   d := 5;
      ...              {position two}
      sub1();
      ...
    end; {sub2}
begin
    a := 1; b := 2;
    ...                {position one}
    sub2();
    ...
end. {test}
```

(1). Determine what variables are visible at positions 1, 2, 3 and what their values are (given the call sequence indicated) with **static scoping**.

(2). Determine what variables are visible at positions 1, 2, 3 and what their values are (given the call sequence indicated) with **dynamic scoping**.

	Static Scoping		Dynamic Scoping	
Position	variable	value	variable	value
one	a		a	
	b		b	
two	a		a	
	b		b	
	c		c	
	d		d	
three	a		a	
	b		b	
	c		c	
	e		d	
			e	