1.a. Choose a programming language. Give an example of a syntactic rule and its associated semantics. (4 points)

b. Define BNF and state its major shortcoming in the description of grammars for programming languages. (4 points)

c. What is the dangling-else problem found in some programming languages? Name one such language and give a way to fix such problem. (6 points)

2a. Consider the languages FORTRAN, C, and Lisp. For each of these give a language characteristic that completely distinguish it from the other two. (6 points)
2b. Consider the programming languages FORTRAN’58, Algol’60, Ada’83, Java; state one new programming construct that the language introduced and was not available before. (4 points)

.2.c. Give two examples of binding that have to be statically bound. (2 points)

d. give two examples of bindings that have to be semi-dynamically bound. (2 points)

e. give two examples of binding that have to be dynamically bound. (2 points)

f. give two examples of bindings that can benefit if the binding is delayed for as late as possible. State the binding time along with the example. (2 points)
g. when is a call to a function bounded to the body of the function to be executed? Answer this question for each procedural programming and Object Oriented programing. (4 points)

3.a. State the most important construct to express computations found in a programming language (2 points)

c. Give an example illustrating the importance of the evaluation order of expressions. (4 points)

d. Knowing that most programming languages do not specify the order of evaluation of expressions, what mathematical property of expressions is lost? and what does this mean to programmer? (2 points)

e. Does the order of evaluation of expressions matter for functional programming languages? Clearly STATE why or why not. (2 points)
4. For each of the 3 different storage semantic models found in programming languages define it, and list its trade-off points. (**12 points**)
5. Consider C with the extra addition of allowing nested functions. For the program given below, develop the stack of activation records; they must clearly show all the data for each activation record, and they must support the final answer the main. (10 points)

```c
int x = 1;
int f1 (int a, int b) {
    int x = 10;
    int f2 (int a) {
        x = a - x;
        return x;
    }
    int f3(int a) {
        int x = 100;
        return a/f2(a);
    }
    int f4(int b){
        return f3(x + b);
    }
    return f4(a+b);
}
void main() {
    f1(0, 10);
    printf(" The answer is>>%d", x);
}
```

Write the answer of the main based on static scope: The answer is>>

Write the answer of the main based on dynamic scope: The answer is>>
6. State a distinct characteristic between each of the following type of arrays: static array, semi-dynamic array, dynamic array. (6 points)

7a. What kind of record structures appear in ANSI/C? (4 points)

b. Are variant records supported in ANSI/C? if so, can they be made secure for the access of fields that may or not may be available at a given time? (3 points)

b. Can programmer rely on the use of records, as given in ANSI/C use to introduce user-defined types? (3 points)

c. What is missing in the use of records in ANSI/C to provide us with a good enough construct to support the software continuity principle and the information hiding principle? (4 points)
8.a. What is polymorphism? (3 point)

b. What is ad-hoc polymorphism? Provide an example of such. (3 point)

c. What is universal polymorphism? (3 point)

d. What type of polymorphism does Java support and implement? (2 point)

9a. List the two fundamental uses of pointers in programming languages (4 points)

b. Are pointers an abstraction mechanism? (1 point)

c. Do languages which do not provide pointers (explicitly or implicitly) fail in providing support to a fundamental construct used by the programmers? (2 points)
10. List by name all the parameter passing mechanisms available in programming languages; and indicate whether the matching argument can be: an expression, a variable or both. (12 points)

Parameter passing mechanism               Matching argument

12. Consider the following code:
```c
int var1 = 5;  // this is a global variable
int funct ( int par1, int par2) {
    ....par1 = par2;
    ....var1 = var1 + 1;
    par2 = par2 + 1;
    return par1 + 1;
}

//client code
int list[5];
list[var1] = 10;
int result = funct(var1, list[var1]);
println("var1 has value >>" + var1 + ".Result is " + result);
```

Complete the following answers: (3 points each)
Using value parameter passing for par1 and par2, the answer is:
var1 has value >> _______.Result is _______

Using result parameter passing for par1 and by-value for par2, the answer is:
var1 has value >> ______.Result is _______

Using value-result parameter passing for par1, and by-value for par2, the answer is:
var1 has value >> ______.Result is _______

Using reference parameter passing for par1 and by-value for par2, the answer is:
var1 has value >> ______.Result is _______
11a. Consider the following code to compare between eager and delayed evaluation:

double func (int x, double y) {
    if (x > 0)
        return (x-y) * (x + y);
    else
        return x*x/(2-x);
}

assuming that for the client: u and v are integer variables with values: u = 10; v = -5.

Evaluate the following call: f(v-u, (u-v)*(u + v)/(v*v)) using eager and delayed (or by-need, not lazy!) evaluation; you must clearly indicate the differences in the evaluation process. (8 points)

b. Will lazy evaluation make any difference in the evaluation of the previous code? Clearly explain using that code. (4 points)
c. In functional programming languages, and with the appropriate evaluation time policy we can define infinite data structures. State the kind of evaluation time policy that can be used to accomplish this and provide a simple example of such an infinite structure. (8 points)

d. How does a user manipulate such an infinite structure as defined above by you? (4 points)

12.a. What parameter passing mechanisms are supported by functional programming languages? (2 points)

b. Does it matter if we change the parameter passing mechanism of functions in a program written in LISP from passing by value to passing by reference? Why or why not? (4 points)

13.a. List the basic components that a run-time system MUST have and state the purpose for each of those components. (6 points)

b. Does the run-time system has to have a Garbage collector? clearly support your answer. (4 points)
15. You wrote an application in 3 different languages: Java, ANSI/C and Scheme. For each of them clearly state the trade-off of writing the solution in such language. Do not include in the trade-offs your inexperience with the language being discussed. (12 points)
16. For each of the languages: ANSI-C, Java, Ada (‘83, ‘95) clearly state the main abstractions and access mechanisms to decompose programs into modules. (12 points)