1a. Define Programming Language. (3 points)

b. From the point of view of development of software packages, list the kinds of activities that a programming language ought to support. (4 points)

b. List the 3 factors in the design of a programming language that can affect its capacity to express computations. (3 points)

c. Give two examples for each factor listed above using the same programming language for all the examples; for each of those examples, one where the construct used affects expressibility in a positive light, one example where the construct affect expressibility in a negative way. (Note: you must give a total of 6 examples). State the language chosen here: __________________ (12 points)
2.a. What is the “cost” of using Java. (6 points)

b. For each decade from the 50’s to the 90’s, name a programming language and state how the language improved the programming language landscape upon its appearance. (4 points)
3. Define programming language paradigm. (2 points)

4. For each of the 4 most common paradigms (procedural, OO, functional and logical) give:
   a. model of computation. (4 answers) (8 points)

   b. Programming methodology supported by the model (i.e. how are programs developed using that paradigm. (4 answers.) (8 points)

   c. Give an example of a language for each paradigm (name paradigm and language, 4 answers.) (4 points)
d. For each paradigm of the following paradigms, OO, functional and logical, name a characteristic that distinguishes from the other two paradigms. (8 points)

5. Evaluate compilers and interpreters in terms of storage and memory usage. (6 points)
6.a. **Define** the methods used for evaluation of expressions as found in programming languages. (4 points)

b. Name a language for which one of the methods is used mostly by that language; state language and method. (2 points)

c. Can a programming language implement only ONE of those methods? Clearly state the method, and the reason why the language need only implement that one. (4 points)

d. Procedural and OO programming languages use only one evaluation method predominantly. Name the method. (1 point)

e. Can every expression in procedural and OO languages be evaluated using that method alone? *If yes,* clearly justify that the method mentioned is sufficient. *If the answer is no,* clearly illustrate via an expression that the stated method could not be used. (3 points)

f. In terms of program efficiency, clearly explain which method is more efficient. (4 points)
7.a. List by name all the parameter passing mechanisms provided in programming languages. and indicate whether the
matching argument can be: an expression, a variable or both. (12 points)

<table>
<thead>
<tr>
<th>Parameter passing mechanism</th>
<th>Matching argument</th>
</tr>
</thead>
</table>

b. Consider the following code:

```java
int var1 = 5;    // global variable for funct given below.

int funct ( int val ){
    var1 = var1 + 1;
    val = val + val;
    return val + 1;
}

//client code
funct(var1);
println(“var1 has value >>” + var1);
```

Complete the following answers: (3 points each)

Using value parameter passing for val the answer is:
var1 has value >> _______

Using result parameter passing for val the answer is:
var1 has value >> _______

Using reference parameter passing for val the answer is:
var1 has value >> _______
c. Give the output of the following program where the parameter of the method `func` are passed each using the same mechanism. Give your answer for the four parameter passing mechanisms by-value, by-reference, by-value-result, by-name:(12 points)

```java
int i = 0;
int table[2] = { 1, 1};

public static void func ( int x, int y){
    x = x + 1;
    i = i + 1;
    y = y + 1;
}

public static void main (String[] args) {
    func(a[i], a[i]);
    println ("a[0] = " + a[0] + " a[1] = " + a[1]);
}
```
8.a. Write a Haskell function \texttt{xOrDiff} taking as input two \texttt{int} lists, and finds the elements that are in either list but not in both. \textit{Example:} \texttt{xOrDiff [1,5,10] [12, 5, 3]} \textit{---}> \texttt{[1,10, 12, 3]}. and \texttt{xOrDiff [1,5, 3] [1, 3]} = \texttt{[5]}, \texttt{xOrDiff [1, 5, 7] [1,5,7]} = \texttt{[]}. Write the type of the function followed by its implementation.\textbf{(6 points)} For extra points, write the function but taking any two lists of the same type of entries.

b. Write selection sort in Haskell, assume sorting in increasing order. Recall: in selection a list of length at least two is given (sorting other cases is trivial), and its smallest element is places in the first position, then the second smallest is place in the second position, and so on. You may assume that the list entries are of any built-in type. Write the type of the function followed by its implementation.\textbf{(8 points)} For extra points, write the sort but the list is of any type of entry, and order is determined by client.