CS324 MIDTERM EXAM Answers of the sample questions

I) TRUE (T) answers are marked T only (not marked questions are by default False).

a) The higher the programming language’s level the faster and more readable is its code.

b) Generally, the addition of polymorphic features in a programming language makes it more powerful. T

c) The parse tree of a compiled program is generated by the syntax-analyzer. T

d) The HLL’s program syntax to semantics mapping defines the HLL code readability and power. T

   When the syntactic description is not verbose, i.e., concise, it is more readable; also when it reveals huge semantics then this is power. Both features are related to syntax semantics mapping.

e) The FORTRAN designers sacrificed the language’s security, abstraction, and power for efficiency. T

f) In dynamic type checking, the CPU uses the program parse-tree to finish the type checking process.

g) The addition of enumerated types to a HLL makes for more secure programming in such language. T

h) Any HLL that has a secure typing system is a secure language.

i) In Pascal, the set of scopes that represents the environment of a name may, or may not, be nested.

j) Recursion forces binding of names to their allocated memory locations at run time. T

k) In statically scoped languages, to access non-local name in a callee, the CPU will scan the static chain to get to the environment of definition of the callee. F

   The CPU follows the "static chain" to get to the AR where the nonlocal is defined, not the definer of the callee (which is the first step in the chain).

l) The compiler increments/decrements its static nesting level (SNL) counter upon entering/existing into/from procedure and/or function, respectively. T

m) Treating functions as first class citizens adds polymorphic power to the hosting HLL. T

n) The environment that is used for name interpretation might be a set of nested “scopes” of declarations. T

o) The static nesting level (SNL) of a name declaration is always lower than the SNL of its use.

p) In statically scoped HLL’s, the CPU might not be able to access a non-local name.

   The CPU must be able to access any nonlocal, since the issue of access is settled at compile time.

q) If the CPU does not find the AR of the callee’s definer, it will allocate it at run time, and proceed.

r) The heap model of computation is used in languages that treat functions as first class citizens (FCC). T

s) Function’s formal parameters are not considered a part of its local names.

t) Java object references are more secure than pointers in C. T

u) The subrange mechanism in Pascal is an insecure feature. T

v) In block structured programs, it is not efficient to deeply nest functions/procedures definitions. T

w) A closure is the value evaluated for an actual parameter of type function/procedure. T
Yes, a procedural formal parameter can not be passed as an actual parameter to another procedure, at the code of the hosting callee. F (we just pass the its closure, which is known by its host)

Functions as FCC is a violation of the contour diagram (CD) concept of information hiding/abstraction. T

Returning a function $F$ from another function, say $A$ to a procedure $H$, allows $H$ to pass $F$ as an actual parameter to another (visible) procedure, say $P$. Now, $P$ can call $F$ which might be not directly visible via the static scoping approach.

The contour diagram (CD) is very useful and often utilized in dynamically scoped HLLs.
MCQ questions: select (circle) the BEST answer:

1) We study HLLs in order to:
   a) have the best language choice to solve a problem  b) find the cost of buying its compiler.
   c) a and future design of a new HLL    d) c and revising all of the code we have       e) none of the above

2) The following factors make for a “good” HLL, regardless of the environment of its usage:
   a) Free of cost ($) and HLL’s popularity  b) HLL security, abstraction, and modularity
   c) a and compiled or interpreted translation   d) b and portability   e) c and d above

3) The notion of power in a HLL is expressed via the following:
   a) recursion     b) the amount of semantics carried by syntax (syntax to semantics mapping)   c) a and b
   d) c and polymorphic features       e) d and code reusability     f) a and secure typing system

4) The following HLLs feature(s) represent(s) a tradeoff between power (gain) versus security (loss):
   a) Algol’s by-name parameter passing mechanism      b) aliasing and global declarations    c) a and b
   d) c and dynamic type checking   f) d and dynamic scoping

5) Early FORTRAN is an example of a HLL that is very:
   a) powerful        b) secure       c) abstract        d) all of the above       e) none of the above

6) In addition to overworking the integer type with label, the following caused a potential security
   loophole in FORTRAN:
   a) implicit typing     b) similar syntax of totally different semantics constructs          c) a and b
   d) global declaration       e) c and ignoring blanks everywhere
   f) none of the above

7) The AR of a callee, say procedure P, may contain:
   a) the return address of the caller of P          b) P’s static nesting level            c) P’s static distance
   d) the return address of the caller of P        e) all of the above          f) none of the above

8) In Pascal, in searching for the memory location of a nonlocal name, in procedure P, the CPU will start
   at:
   a) the AR pointed to by of the dynamic link (DL) pointer in P’s AR       b) P's AR
   c) the AR of the main-program        d) P's caller of the caller AR       f) none of the above

9) Some of the major new features that Algol included and did not exist in early FORTRANs:
   a) recursion, dynamic arrays, by-name parameters, blocks, and free-format
   b) global variable declarations, nesting of scopes, and compound statements
   c) powerful structuring constructs (e.g., the for, switch, and if statements)
   d) dynamic and static scoping, and pass by-constant       e) all of the above
   f) a, b, c above and dynamic binding of names to memory locations     g) f and the contour diagram

10) Following is the output of the “print (i)” in the Algol for statement:
    for i := 1, 3, 6 step 1 until 8, i/2 while i ≥ 1 , 2 step i until 4 do print (i);
    a) 1, 3, 6, 7, 8, 4, 2, 1, 2, 4              b) 1, 3, 6, 7, 8, 4, 2, 2, 3, 4
    d) 1, 3, 7, 8, 4, 2, 2, 4                      e) none of the above

11) The Backus-Naur Form (BNF) is:
    a) a meta-language equivalent to a CFG and more powerful than a regular grammar (RG)
    b) it describes only the syntax of other HLLs      c) a and b above
    d) an executable meta-language equivalent to context free grammar (CFG)
    e) b above and it might be able to capture only some of the typing semantics of a HLL
    f) none of the above
12) The following language issue(s) can be described via BNF:
   a) statement sequence   b) matching parenthesis   c) undeclared name   d) operator associativity
   e) all of the above  f) a, b, and d above  g) f and over/underflow of values

13) Given the following BNF for expression:
   
   \[
   \begin{align*}
   \text{<expr>} & ::= \text{<expr>} + \text{<term>} | \text{<expr>} - \text{<term>} | \text{<term>} \\
   \text{<term>} & ::= \text{<term>} \ast \text{<factor>} | \text{<term>} \div \text{<factor>} | \text{<factor>} \\
   \text{<factor>} & ::= \text{id} | \text{number} | (\text{<expr>})
   \end{align*}
   \]
   
   The following is a correct expression based on the above BNF (assume \text{id} & \text{number} as known):

   a) \([ \text{X/Y} \ast 8]\)  
   b) \(7 \ast (8 + 5)\)  
   c) \(7 (\text{X + Y})/2\)  
   d) \((7 (\text{X+Y}) )\)  
   d) all of the above  
   f) none of the above

14) The following features are potential threats to Algol’s security (security loophole):
   a) by-name   b) dynamic arrays   c) variant records   d) blocks
   e) the 0-1-∞ design principle  f) all of the above

15) The Algol \textit{by-name} and Pascal \textit{by-reference} are the same in case of the \textit{actual} parameter is:
   a) invoked in the environment of its callee   b) a complex expression   c) a and b above
   d) a function call (as in the Jensen’s device)  c) none of the above