COMPILER DESIGN TEST 2

Number of Questions: 25 **Section Marks: 30**

Directions for questions 1 to 25: Select the correct alternative from the given choices.

1. Which of the following issues are considered by the run-time environment of a compiler?
(i) Allocation of storage locations for the objects
(ii) Mechanisms to access variables
(iii) Linkages between procedures
(A) (i), (ii) only (B) (ii), (iii) only
(C) (iii) only (D) (i), (ii), (iii)

2. Program data objects (such as global constants) and data generated by the compiler (such as the information to support garbage collection) are placed in which of the following areas of run-time memory?
(A) Code area (B) Static area
(C) Heap area (D) Stack area

3. If an activation of procedure '*A*' calls procedure '*B*' then which of the following is TRUE?
(A) Activation of *B* must end before the activation of *A* can end.
(B) Activation of *A* must end before the activation of *B* can end.
(C) Activation of *A* must end before the activation of *B* can start.
(D) Activation of *B* must start after the activation of *A* can end.

4. Consider the activation of procedure *X* which calls procedure *Y*. the activation of *Y* aborts (i.e., it becomes impossible for execution to continue). Then which of the following is TRUE?
(A) *X* continues execution
(B) *X* ends simultaneously with *Y*
(C) *X* makes '*Y*' to execute
(D) None of these

5. Which of the following is true with respect to an "activation tree"?
(i) Root of the tree is the activation of "main" procedure.
(ii) Nodes of the tree corresponds to an activation.
(iii) At a node for an activation of procedure *P*, the children correspond to activations of the procedures called by this activation of *P*.
(A) (i), (ii), only (B) (ii) only
(C) (iii) only (D) (i), (ii), (iii)

6. The sequence of procedure calls of a program corresponds to which traversal of the activation tree?
(A) In order traversal (B) Pre order traversal
(C) Post order traversal (D) Level-order traversal

7. The sequence of returns of procedures corresponds to which traversal of the activation tree?
(A) In order traversal
(B) Pre order traversal
(C) Post order traversal
(D) Level-order traversal

8. Consider an activation tree '*T*'. Also assumes that the control lies within a particular activation of some procedure, corresponding to a node *N* of the activation tree.
Then the activations that are currently live (or open) are those that correspond to
(A) Node *N* only
(B) Node *N* and its ancestors only
(C) Node *N* and its children only
(D) Ancestors of node *N* only

9. Which of the following tasks is managed by a control stack (i.e., run-time stack)?
(A) Static data and functions
(B) Garbage collection
(C) Procedure calls and returns
(D) All of the above

10. Which of the following field of an activation record will point to the activation record of the caller?
(A) Returned values (B) Access link
(C) Temporaries (D) Control link

11. Consider the following statement :
"It is possible to allocate objects, arrays or other structures of unknown size on the stack".
Which of the following is TRUE with respect to given statement?
(i) This avoids the expense of garbage collection.
(ii) The array or the object needs to be local of a procedure.
(A) (i) only (B) (ii) only
(C) Both (i) and (ii) (D) Neither (i) nor (ii)

12. Consider the languages which do not allow nested procedure declarations. Then which of the following is FALSE?
(i) Global variables are placed on stack.
(ii) Access links are used for accessing local variables of a procedure.
(A) (i) only (B) (ii) only
(C) Both (i) and (ii) (D) Neither (i) nor (ii)

13. Consider the following procedure:

```
q( )
{
.
.
.
p ( )
```

```
{  .
 .
 .
.
r( )
{
 .
 .
 }
s( )
{  .
 .
 .
}
 }
 t( )
 {
 .
 .
 .
 }
 }
```

What is the nesting depth of procedure '*s*'?

(A) 2 (B) 3
(C) 4 (D) 5

14. Which of the following is attached to each activation record to use normal static scope rule for nested functions?

(A) Access link (B) Control link
(C) Process link (D) Data link

15. Consider a procedure '*P*' which is at the top of the stack with nesting depth 'n_p' and '*P*' needs to access '*x*', which is defined with in some procedure '*Q*' that surrounds '*P*' and has nesting depth 'n_q'. To find '*x*' initially top of the stack is checked and then how many times access links are followed?

(A) $n_q - n_p$ (B) $n_p - n_q$
(C) $n_q - n_p + 1$ (D) $n_p - n_q + 1$

16. Consider a procedure '*p*' which is passed to another procedure '*q*' as a parameters. And *q* calls its parameter. Then which of the following things are passed from caller to called?

(i) Name of the procedure parameter
(ii) Access link of the parameter

(A) (i) only (B) (ii) only
(C) Both (i) and (ii) (D) None of these

17. Which of the following is FALSE with respect to reference counting?

(i) The reference count of a new object is set to 1.
(ii) The reference count of each object passed into a procedure is incremented.
(iii) Reference counting can be used to collect unreachable cyclic data structures.

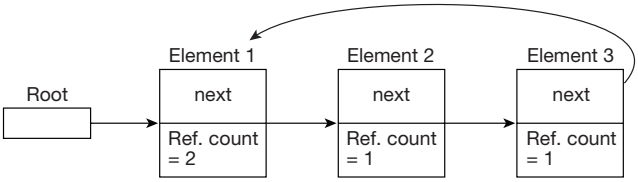
(A) (ii) only (B) (iii) only
(C) (ii), (iii) only (D) (i), (ii), (iii)

18. Which of the following statements is TRUE?

(i) If no references are found to an object then that object is considered as 'garbage'.
(ii) Reference counting garbage collector works in 'incremental' fashion.

(A) (i) only (B) (ii) only
(C) Both (i) and (ii) (D) Neither (i) nor (ii)

19. Consider the following figure:



Each object has its Reference count. If the 'Root' assigned 'NULL' value then which of the following is true?

(i) The reference counts of element 1, 2, 3 will be respectively 1, 1, 1.
(ii) The reference counting garbage collector can consider the elements as garbage.

(A) (i) only
(B) (ii) only
(C) Both (i) and (ii)
(D) Neither (i) nor (ii)

20. Which of the following task is not related to code generator?

(i) Instruction selection
(ii) Register Allocation
(iii) Instruction ordering

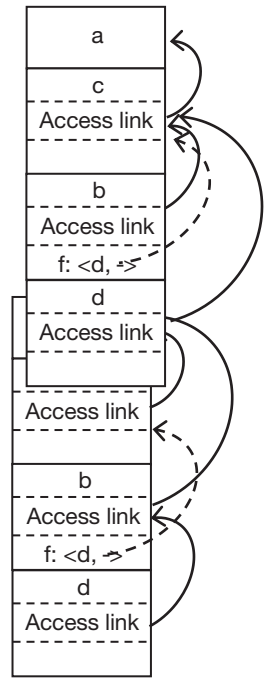
(A) (i), (ii) only (B) (ii), (iii) only
(C) (iii) only (D) None of these

21. Consider the following function:

```
fun a(x) =
    Let
       fun b(f) =
       ….f….;
       fun c(y) =
         Let
       fun d(z) = ……
       in
       …..b(d)…….
       end
    in
    ……..c(1)……
    end;
```

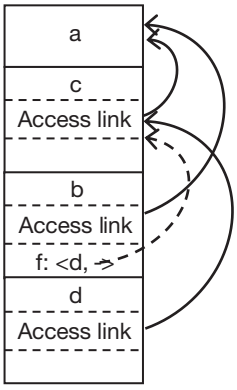
Here function '*a*' consists the functions '*b*' and '*c*' within it. Function '*b*' has function-valued parameter *f*, which is called by '*b*'. Function '*c*' defines within it self a function '*d*' and then '*c*' calls '*b*' with actual parameter '*d*'. Then which of the following correctly shows the access links between the functions?

(A)

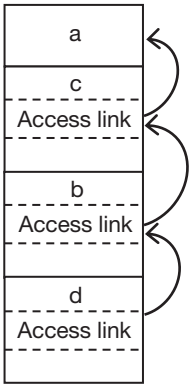


(B)

(C)



(D)



22. The number of non-leaf nodes present in the DAG representation of the following block is:

1. $x = y + z$
2. $y = x - w$
3. $z = y + z$
4. $w = x - w$

(A) 2 (B) 3
(C) 4 (D) 5

23. Consider the following block:

1. $p = q + r$;
2. $q = q + s$;
3. $r = r - s$;
4. $t = q + r$;

Which of the following is TRUE with respect to given block?

(i) DAG representation of given block can identify the common sub-expression of the block.
(ii) The DAG has 4 non-leaf nodes
(A) (i) only
(B) (ii) only
(C) Both (i) and (ii)
(D) Neither (i) nor (ii)

24. Consider the following expression:
$(c + d) - (-(f + g))$
To evaluate the given expression (All operands must be in registers). What is the minimum number of registers required?

(A) 1 (B) 2
(C) 3 (D) 4

25. Construct a syntax-directed translation scheme that takes strings of *a*'s, *b*'s and *c*'s as input and produces as output the number of substrings in the input string that correspond to the pattern $a(a + b)^*c + (a + b)^*b$. For example, the input string '*abbcabcababc*' translation scheme outputs 3(*abbc*, *abc*, *ababc*).

(A) $S_1 \rightarrow S_2 a \{S_1.n_1 = S_2..n_1 + 1;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow S_2 b \{S_1.n_1 = S_2.n_1 + 1;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3 + S_2.n_2;\}$
$S_1 \rightarrow S_2 c \{S_1.n_1 = 0;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow a \{S_1.n_1 = 1;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$
$S_1 \rightarrow b \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0\}$
$S_1 \rightarrow c \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0\}$
(n_1 = count of number of *a*'s to the left of *c*
n_2 = count of number of *a*'s to the right of '*c*',
n_3 = total number of required substrings.)

(B) $S_1 \rightarrow S_2 a \{S_1.n_1 = S_2.n_1 + 1;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow S_2 b \{S_1.n_1 = S_2.n_1 + 1;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow S_2 c \{S_1.n_1 = 0;$

$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow a \{S_1.n_1 = 1;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$
$S_1 \rightarrow b \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0\}$
$S_1 \rightarrow c \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$

(C) $S_1 \rightarrow S_2 a \{S_1.n_1 = S_2.n_1 + 1;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow S_2 b \{S_1.n_1 = S_2.n_1;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow S_2 c \{S_1.n_1 = 0;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow a \{S_1.n_1 = 1;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$
$S_1 \rightarrow b \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$
$S_1 \rightarrow c \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$

(D) $S_1 \rightarrow S_2 a \{S_1.n_1 = S_2.n_1 + 1;$
$S_1.n_2 = S_1.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow S_2 b \{S_1.n_1 = S_2.n_1;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow S_2 c \{S_1.n_1 = 0;$
$S_1.n_2 = S_2.n_2;$
$S_1.n_3 = S_2.n_3;\}$
$S_1 \rightarrow a \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$
$S \rightarrow b \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$
$S \rightarrow c \{S_1.n_1 = 0;$
$S_1.n_2 = 0;$
$S_1.n_3 = 0;\}$

Answer Keys

1. D	**2.** B	**3.** A	**4.** B	**5.** D	**6.** B	**7.** C	**8.** B	**9.** C	**10.** D
11. C	**12.** C	**13.** B	**14.** A	**15.** B	**16.** C	**17.** B	**18.** C	**19.** A	**20.** D
21. C	**22.** B	**23.** B	**24.** C	**25.** A					

Hints and Explanations

1. The run-time environment of a complier deals with the issues such as:
(1) Allocation of storage locations for the objects
(2) Linkages between procedures
(3) Mechanisms used by the target program to access variables
(4) Mechanisms for passing parameters etc.
Choice (D)

2. The size of some program data objects and data generated by the compiler can be known at compile time. So they can be placed in static area of run-time memory.
Choice (B)

3. Given, that *A* and *B* are procedures and

```
A ( )
{
.
.
B ( ) ;
.
.
}
```

Each time a procedure is called, space for its local variables is pushed on to a stack and when the procedure terminates, that space is popped off the stack.
In given scenario, Activation record of '*A*' created initially. Next *B*'s activation record is created and *B* must end before the termination of '*A*'. Choice (A)

4. If *Y* aborts then *X* also ends simultaneously with *Y*.
Choice (B)

5. Choice (D)

6. The preorder traversal of an activation tree corresponds to the sequence of procedure calls. Choice (B)

7. The sequence of returns corresponds to a post-order traversal of the activation tree. Choice (C)

8. The live nodes are *N* and its ancestors. (∵ After return of child nodes only parent nodes will terminate.)
Choice (B)

9. Control stack (run-time stack, manages procedure calls and returns. Choice (C)

10. Control link points to the activation record of the caller. Access link is used to locate data needed by the called procedure. Choice (D)

11. It is possible to allocate objects, whose size cannot be determined at compile time on a stack. This avoids the expense of garbage collecting their space.
But the stack can be used only for an object if it is local to a procedure. ($\because$ The local data will be inaccessible when the procedure returns). Choice (C)

12. Global variables are allocated in static storage.
Local variables are placed in stack (and accessed using top of the stack). Choice (C)

13. Nesting depth of a procedure is '1' if it is not nested within any other procedure.
If a procedure P is defined immediately with in a procedure Q then the nesting depth of $P = 2$
Here, in given code,
Nesting depth of 'q' = 1
Nesting depth of 'p = 2
Nesting depth of 'r' = 3
Nesting depth of 's' = 3
Nesting depth of 't' = 2 Choice (B)

14. Access links are attached to activation records to use static scope rules for nested procedures. Choice (A)

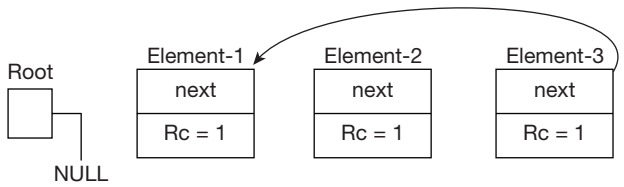
15. P is defined with in Q. Nesting depth of P is n_p. Nesting depth of 'Q' is n_q.
As P is inside Q,
$N_p \geq n_q$
We need to check $n_p - n_q + 1$ records to identify 'x'.
(one access is already given in problem). Choice (B)

16. The name of the procedure parameter and access link of the parameters needs to passed by caller. Choice (C)

17. Reference counting cannot collect unreachable, cyclic data structures. Choice (B)

18. An object is considered as garbage if it has no references.
References counting garbage collector works in an 'incremental' fashion. Choice (C)

19. The reference count of an object holds a value which specifies the number of objects referencing that object.
If root is made as "NULL : THEN:

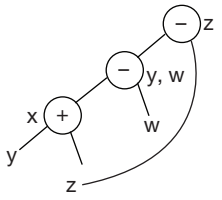


$\therefore$ (i) is correct
But References counting garbage collector cannot consider these elements as garbage ($\therefore$ their Reference counts is not zero).
$\therefore$ (ii) is false. Choice (A)

20. Choice (D)

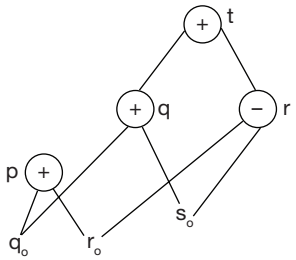
21. Let us see what will happen when 'a' is executed. First 'a' calls 'c', so we place an activation record for 'c' above that for 'a' on the stack. The access link for 'c' points to the record for 'a', since 'c' is defined immediately within 'a'. Then 'c' calls $b(d)$.
'c' knows about 'd', since 'd' is defined within 'c' and therefore 'c' passes a pointer to its own activation record as the access link. Choice (C)

22. Given block
$x = y + z$
$y = x - w$
$z = y + z$
$w = x - w$



Number of non-leaf nodes = 3 Choice (B)

23. Given block,
$p = q + r$
$q = q + s$
$r = r - s$
$t = q + r$
This block calculates '$q + r$' two times but both times The sum '$q + r$' is same. But the DAG does not consider this issue.
$(t = q + r = (q + s) + (r - s) = q + r)$
The DAG of given code will be



Number of non-leaf nodes = 4 Choice (B)

24. $R_0 \leftarrow c$
$R_1 \leftarrow d$
$R_0 \leftarrow R_0 + R_1$
$R_1 \leftarrow f$

$R_2 \leftarrow g$
$R_1 \leftarrow R_1 + R_2$
$R_1 \leftarrow -R_1$
$R_0 \leftarrow R_0 - R_1$
$\therefore$ 3 registers required.

Choice (C)

25. The CFG for given problem is

$$S \rightarrow Sa|Sb|Sc|a|b|c$$

This grammar parses any string in left-skewed manner. i.e., grammar is left-recursive. Lets take three synthesized attributes for the non-terminal symbol S, namely n_1, n_2, n_3. n_1 will capture the number of a's to the left of a given c, n_2 will count the number of a's to the right of given 'c' character and n_3 will accumulate the total number of substrings. Here we need to count the number of a's to the left of a 'c' character and to the right of that character so that we can add the value of n_1 to a running total for each occurrences of a 'b' character to the right of 'c' which recording the value of a's to the right of 'c' so that when we find a new 'c', we copy the value of the a's that were to the right of the first 'c' and which are now to the left of the second 'c'.
Hence, the resultant grammar is Choice (A).

Choice (A)