SASTRA DEEMED TO BE UNIVERSITY (U/S 3 OF THE UGC ACT, 1956) — THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

School of Computing
I B.Tech. CSBS
First CIA Test – January 2020
Course Code: CSE209
Course Name: Data Structures and Algorithms
Duration: 90 minutes **Max Marks: 50**

PART A 10 x 2 = 20 Marks

Answer ALL questions

1. Consider the following algorithm:

Algorithm A

```
1. i = 1
2. while (i < 10)
3.     print "i = ", i
4. return
```

Is this algorithm correct? Which characteristics (properties) of algorithm are lacking?

Ans.: The algorithm lacks finiteness property.

2. Find the time complexity of the following algorithm to find the last occurrence of the minimum element in array A:

Algorithm MIN_INDEX(A)

```
1. min_idx = 1
2. for i = 2 to A.length
3.     if A[i] <= A[min_idx]
4.         min_idx = i
5. return min_idx
```

Ans.: T(n) = $\Theta(n)$

3. Consider the following two algorithms to find the sum of all numbers of an array A.

Algorithm SUM(A,n)

```
1. sum = 0
2. for i = 1 to n
3.     sum = sum + A[i]
4. return sum
```

Algorithm RSUM(A,n)

```
1. if n = 0
2.     return 0
3. else
4.     return RSUM(A, n-1) + A[n]
```

Find the time complexity for both of these algorithms. Which algorithm is faster? Justify your answer.

Ans.: $T_{SUM}(n) = \Theta(n)$ and $T_{RSUM}(n) = \Theta(n)$. Even though both are having same time complexity, since Algorithm SUM is iterative it is more faster than Algorithm RSUM which has additional recursive overhead.

4. Write the recursive algorithm for finding nth number in the Fibonacci sequence.

Ans.: Algorithm FIB(n)

```
1. If n=1
2.     return 0
3. else if n=2
```

```
4.     return 1
5. else
6.     return FIB(n-1) + FIB(n-2)
```

5. “Postfix expressions are unambiguous”. Justify.
   Ans.: Postfix expressions are parentheses free expressions and the operators are placed immediately after their left and right operands. Hence they are unambiguous and easy to evaluate.

6. Convert the following infix expression to postfix expression: a+b*(c+d)/(e-f)-g
   Ans.: a b c d + * e f - / + g –

7. Consider the following function calls in a C program:

```
void fun1(int x)   {
    printf(“In fun1() before calling fun2() -- x = %d”, x);
    fun2(x);
    printf(“In fun1() after calling fun2() – x = %d”, x);
}
void fun2(int x)  {
    x = 2 * x;
    printf(“In fun2() before calling fun3() -- x = %d”, x);
    fun3(x);
    printf(“In fun2() after calling fun3() – x = %d”, x);
}
void fun3(int x)  {
    x = 2 * x;
    printf(“In fun3() -- x = %d”, x);
}
```

   Which data structure is used by the system for implementing function calls? Why?
   Ans.: Stack data structure is used by the system to store the current values of local variables, parameters and the return address. When returning from a function call the control should transfer to the most recent function that called this function. As it follows Last-In-First-Out kind of returns from functions, stack data structure is appropriate for it.

8. Evaluate the following postfix expression using appropriate data structure:

   a b + c * d – e c / b a – + *

   where a = 1, b = 2, c = 3, d = 4, e = 6
   Ans.: Result = 15.   1+2=3;   3*3=9; 9-4=5;   6/3=2; 2-1=1;   2+1=3; 5*3=15

9. Write the algorithm for inserting an element into a queue.
   Ans.: Algorithm ENQUEUE(Q, x)

```
1. if ISFULL(Q)
2.     print “Queue Overflows”
3.     return
4. else
```

```
5.    Q.rear = Q.rear + 1
6.    Q[Q.rear] = x
7.    if front = 0
8.        front = 1
```

Algorithm ISFULL(Q)

```
1.  if  Q.rear >= Q.length
2.      return 1
3.  else
4.      return 0
```

10. Why circular queue is advantageous than a linear queue?
Ans.: In case of linear queue, once the queue becomes full, till all the elements are deleted from the queue, it is not possible to insert any more element even though few elements from the front of the queue are deleted and empty spaces are available at the front.

PART B **3 x 10 = 30 Marks**

Answer ALL questions.

11. Consider you have been asked to store a lower triangular matrix T of order 1000x1000 in a mobile device having limited memory. Hence you are storing only the non-zero elements in order in a one dimensional array A. For example T[1..3,1..3] is stored in A[1..6] as follows:

$$\overset{T}{\begin{bmatrix}3 & 0 & 0\\6 & 7 & 0\\1 & 4 & 5\end{bmatrix}} \quad \Longrightarrow \quad \overset{A}{\begin{array}{|c|c|c|c|c|c|}\multicolumn{1}{c}{1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5} & \multicolumn{1}{c}{6}\\\hline 3 & 6 & 7 & 1 & 4 & 5\\\hline\end{array}}$$

In such case, write an algorithm that takes i, j where j<=i as input parameter and returns the corresponding element t_{ij} from the array A.
Ans.: In row i, there are i elements. Hence the first element of ith row will be at the position i(i-1)/2+1 in the one dimensional array A. Hence (i,j)th element of T will be in i(i-1)/2+j.
Algorithm RETRIEVE(A, i, j)

```
1.  return A[i*(i-1)/2+j]
```

12. Two stacks are implemented using a single array, which are growing in opposite directions, having two top pointers holding the index of the current top most element of each stack. Write ISFULL, ISEMPTY, PUSH, POP, and PEEK algorithms that also includes the stack number as one of the input parameter to perform insert, delete and retrieve operations on the corresponding stack.

Ans.: Initial value for S.Top1 is 0 and S.Top2 is S.length+1

Algorithm ISFULL(S)

```
1. if S.Top1 = S.Top2 -1
2.     return 1
3. else
4.     return 0
```

Algorithm ISEMPTY(S, n)

```
1. if n = 1
2.     if S.Top1 = 0
3.         return 1
4.     else
5.         return 0
6. else
7.     if S.Top2 = S.length
8.         return 1
9.     else
10.        return 0
```

Algorithm PUSH(S, n, x)

```
1. if ISFULL(S)
2.    print "Stack Overflows"
3. else
4.    if n=1
5.        S.Top1 = S.Top1 + 1
6.        S[S.Top1] = x
7.    else
8.        S.Top2 = S.Top2 - 1
9.        S[S.Top2] = x
```

Algorithm POP(S, n)

```
1. if ISEMPTY(S, n)
2.      print "Stack underflows"
3.      return 0
4. else
5.     if n = 1
6.         x = S[S.Top1]
7.         S.Top1 = S.Top1 - 1
8.     else
9.         x = S[S.Top2]
10.        S.Top2 = S.Top2 + 1
11.    return x
```

Algorithm PEEK(S, n)

```
1. if ISEMPTY(S, n)
2.      print "Stack underflows"
3.      return 0
4. else
5.     if n = 1
6.         x = S[S.Top1]
7.     else
8.         x = S[S.Top2]
9.     return x
```

13. Let Q be an empty circular queue of size 5. Write the values of front and rear along with the contents of Q after each of the following operations:
    i. Enqueue 15, 35, 26, 8, 23, 17
    ii. Dequeue 3 elements
    iii. Enqueue 68, 75
    iv. Dequeue 1 element
    v. Enqueue 51, 82, 93
    vi. Dequeue 3 elements
    vii. Enqueue 56
    viii. Dequeue 4 elements
    ix. Enqueue 11, 22, 33
    x. Dequeue

Ans.:

i. Enqueue 15, 35, 26, 8, 23.

Initial values: front = rear = 0

front = 1 rear = 5 Q

15	35	26	8	23

Enqueue 17: Queue Overflows

ii. Dequeue 3 elements : Returns 15, 35, and 26 for each dequeue in order

front = 4 rear = 5 Q

			8	23

iii. Enqueue 68, 75:
front = 4 rear = 2 Q

68	75		8	23

iv. Dequeue 1 element : Returns 8
front = 5 rear = 2 Q

68	75			23

v. Enqueue 51, 82, 93
Enqueue 51, 82:
front = 5 rear = 4 Q

68	75	51	82	23

Enqueue 93: Queue Overflows;

vi. Dequeue 3 elements: returns 23, 68, and 75
front = 3 rear = 4 Q

		51	82	

vii. Enqueue 56
front = 3 rear = 5 Q

		51	82	56

viii. Deqeueue 4 elements :
For first 3 Dequeue: returns 51, 82, and 56
front = rear = 0 Q

For next Dequeue: Queue Underflows

ix. Enqueue 11, 22, 33
front = 1 rear = 3 Q

11	22	33		

x. Dequeue: Returns 11
front = 2 rear = 3 Q

	22	33		