School of Computing
First CIA Exam – March 2023

Course Code: CSE209 Course Name: Data Structures & Algorithms

Duration: 90 minutes Max Marks: 50

PART A

Answer all the questions **(10 x 2 = 20)**

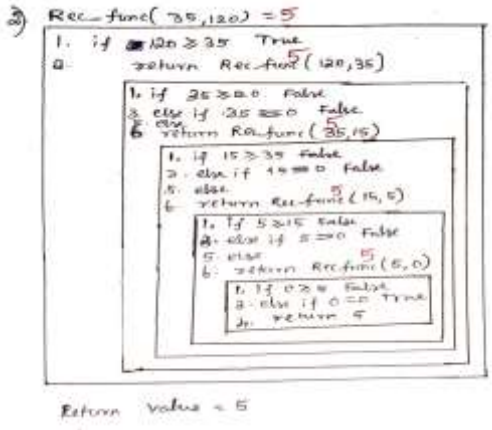
1. Find the bound for the following algorithm:

   $\boldsymbol{Algorithm\ func1(n)}$
   1. $i = 1, sum = 0$
   2. $x = 3, y = 5$
   3. $while\ i \leq n$
   4. $\quad if\ sum\%2 = 0$
   5. $\quad\quad sum = sum + y$
   6. $\quad else$
   7. $\quad\quad sum = sum + x$
   8. $\quad end\ if$
   9. $\quad i = i * 2$
   10. $end\ while$
   11. $return\ sum$

   Ans.: Initial value of i is 1 and it is multiplied by 2 in each iteration of the loop. The loop terminates when i exceeds n. Hence the loop iterates for $\log_2 n$ times. Hence the bound is $\boldsymbol{O(\log_2 n)}$**.**

2. Trace the following recursive algorithm for $m = 35\ and\ n = 120$.

   $\boldsymbol{Algorithm\ Rec\_func(m, n)}$
   1. $if\ n \geq m$
   2. $\quad return\ Rec\_func(n, m)$
   3. $else\ if\ n = 0$
   4. $\quad return\ m$
   5. $else$
   6. $\quad return\ Rec\_func(n, m\%n)$
   7. $end\ if$



3. Rearrange the following functions in the increasing order of their order of growth: $100n^5,\ 35\log_2 n,\ n\log_2 n,\ n^2\log_2 n,\ 4n,\ 2^n,\ 300n^2,\ n!,\ 4^n$

   Ans.: $\boldsymbol{35\log_2 n, 4n, n\log_2 n, 300n^2, n^2\log_2 n, 100n^5, 2^n, 4^n, n!}$

4. Identify the basic operation in the following algorithm and determine how many times it is repeated for $n = 64$. Also find the return value.

   $\boldsymbol{Algorithm\ find\_sum(n)}$
   1. $i = n,\ sum = 0$
   2. $while\ i \geq 1$
   3. $\quad sum = sum + i$
   4. $\quad i = i/2$
   5. $end\ while$
   6. $return\ sum$

   Ans.: Step 3 is the basic operation. For n=64, the loop executes 7 times and the return value is 127.

5. Consider the following matrix of order 4x6. It is stored as 2-D array A in row major order starting from location 2000 and assume that each element is stored using 2 bytes. Find the address of the element A(3,5)=35 will be stored using the formula.

$$\begin{pmatrix} 1 & 40 & 9 & 2 & 4 & 29 \\ 8 & 27 & 16 & 8 & 21 & 1 \\ 3 & 30 & 2 & 3 & \mathbf{35} & 60 \\ 22 & 5 & 7 & 20 & 24 & 17 \end{pmatrix}$$

   Ans.: A(3,5) will be at [2000+ (2*6+4)*2=] 2032

6. Write the recursive algorithm for finding the sum of n elements of a given array.

   Ans.:

   $\boldsymbol{Algorithm\ FIND\_SUM(A, n)}$
   1. $\boldsymbol{if\ n = 0}$
   2. $\quad \boldsymbol{return\ 0}$
   3. $\boldsymbol{else}$
   4. $\quad \boldsymbol{return\ A[n] + FIND\_SUM(A, n-1)}$

7. Consider the following operations on an empty stack of size 3. What will be the contents of stack after each operation?

   i. Push X
   ii. Push V
   iii. Push M
   iv. Push R
   v. Pop
   vi. Pop
   vii. Push E
   viii. Push K
   ix. Pop
   x. Pop

   Ans.: Stack contents after each operation:

3			**M**	**M**				**K**		
2		**V**	**V**	**V**	**V**		**E**	**E**	**E**	
1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
op.	**i**	**ii**	**iii**	**iv**	**v**	**vi**	**vii**	**viii**	**ix**	**x**
top	**1**	**2**	**3**	**3**	**2**	**1**	**2**	**3**	**2**	**1**

8. Assume $a = 5, b = 7, c = 10, d = 18, e = 6$. Evaluate the following postfix expression using stack: $abc * de/+*$. Draw the stack content after each operation.

Next char.	a	b	c	*	d	e	/	+	*
Action	Push	Push	Push	Mul	Push	Push	Div	Add	Mul
4						6			
3			10		18	18	3		
2		7	7	70	70	70	70	73	
1	5	5	5	5	5	5	5	5	365

9. Write the algorithm for pushing all the elements of a file into a stack.

Ans.:

Algorithm MULTI_PUSH($S, file$)

// To insert all elements in given input file into stack S

1. ***open the file in input mode***
2. ***while end of file is not reached***
3. ***Read x from file***
4. ***if top >= MAX_SIZE***
5. ***Print "Stack Overflows"***
6. ***Exit***
7. ***else***
8. ***top = top + 1***
9. ***S[top] = x***
10. ***end if***
11. ***end while***
12. ***close the file***

10. The queue of size 5 is shown below. Draw the contents of the queue after performing Enqueue(72), Dequeue, Enqueue(37), Dequeue operations along with front and rear values.

Q	35	11	45	32		f=1, r=4

Ans.:

	1	2	3	4	5		
Operations \ Q	**35**	**11**	**45**	**32**		**f=1, r=4**	
Enqueue(72)	**35**	**11**	**45**	**32**	**72**	**f=1, r=5**	
Dequeue		**11**	**45**	**32**	**72**	**f=2, r=5,**	**Del. 35**
Enqueue(37)		**11**	**45**	**32**	**72**	**f=2, r=5,**	**Overflow**
Dequeue			**45**	**32**	**72**	**f=3, r=5,**	**Del. 11**

PART B

Answer all the questions **(2 x 15 = 30)**

11. Write the non-recursive algorithm using stack for finding the minimum number of movements required to move N discs of increasing diameter from source needle A to destination needle B using intermediate needle C with the following two constraints:
    (i) Only one disc can be moved at a time and placed in any one of the needles.
    (ii) A larger diameter disc should not be placed on top of a lower diameter disc at any point of time during the movement.

Ans.:

Algoritnm HANOI(N)

// To move N discs from 'A' to 'B' using 'C'.

1. ***x = (N,'A','B','C',13)***
2. ***PUSH(S, x)***
3. ***if $x.N$ = 0***
4. ***goto step $x.RA$***
5. ***x = ($x.N - 1, x.SN, x.IN, x.DN$, 7)***
6. ***goto step 2***
7. ***x = PEEK(S)***
8. ***Print 'Move Disc', $x.N$, ' from ', $x.SN$, ' to ', $x.DN$***
9. ***x = ($x.N - 1, x.IN, x.DN, x.SN$, 11)***
10. ***goto step 2***
11. ***x = POP(S)***
12. ***goto step $x.RA$***
13. ***Return***

12. Convert the following infix expression into postfix expression using stack: $(d * (k + t)/f - u/(r + x) - h)$. Write the contents of stack for each iteration.

Ans.:

Char (x)	Action	Stack Contents	Postfix Exp
(	Push (	(	–
d	Concat d	(	d
*	Push *	(*	d
(	Push (	(* (	d
k	Concat k	(* (	dk

Char (x)	Action	Stack Contents	Postfix Exp
+	Push +	(* (+	dk
t	Concat t	(* (+	dkt
)	Pop +,)	(*	$dkt+$
/	Pop *, & Push /	(/	$dkt+*$
f	Concat f	(/	$dkt+*f$
-	Pop / & Push -	(-	$dkt+*f/$
u	Concat u	(-	$dkt+*f/u$
/	Push /	(- /	$dkt+*f/u$
(	Push (	(- / (	$dkt+*f/u$
r	Concat r	(- / (	$dkt+*f/ur$
+	Push +	(- / (+	$dkt+*f/ur$
x	Concat x	(- / (+	$dkt+*f/urx$
)	Pop +,)	(- /	$dkt+*f/urx+$
-	Pop /,- & Push -	(-	$dkt+*f/urx+/-$
h	Concat h	(-	$dkt+*f/urx+/-h$
)	Pop -,)		$dkt+*f/urx+/-h-$

Postfix Expression: $dkt+*f/urx+/-h-$