| | |
|---|---|
| SASTRA DEEMED TO BE UNIVERSITY (U/S 3 of the UGC Act, 1956) THINK MERIT \| THINK TRANSPARENCY \| THINK SASTRA THANJAVUR \| KUMBAKONAM \| CHENNAI | **School of Computing**<br>**Second CIA Exam – May 2022**<br>Course Code: CSE209 Course Name: Data Structures & Algorithms<br>Duration: 90 minutes Max Marks: 50 |

## PART A

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

1. Write an algorithm to insert an element into linked queue

***Algorithm*** $\boldsymbol{ENQUEUE(FRONT, REAR, x)}$

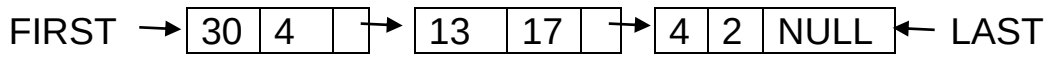
1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow link = NULL$
4. $if\ FRONT = NULL$
5. $\quad FRONT = REAR = T$
6. $else$
7. $\quad REAR \rightarrow link = T$
8. $\quad REAR = T$
9. $return$

2. Write an algorithm to insert an element into beginning of a circular singly linked list.

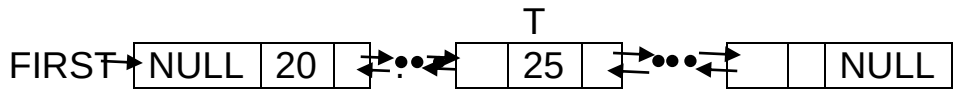
***Algorithm*** $\boldsymbol{INSERT\_AT\_BEG\_CSLL(FIRST, LAST, x)}$

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow link = NULL$
4. $if\ FIRST = NULL$
5. $\quad T \rightarrow link = T$
6. $\quad FIRST = LAST = T$
7. $else$
8. $\quad T \rightarrow link = FIRST \rightarrow link$
9. $\quad LAST \rightarrow link = T$
10. $\quad FIRST = T$
11. $return$

3. Draw the singly linked list representation for the following polynomial: P = $4x^{30}+17x^{13}+2x^4$



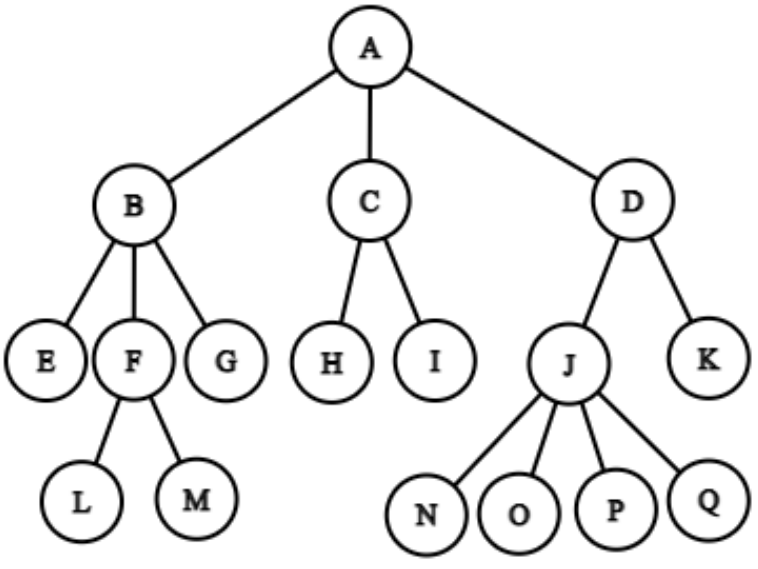
4. Let T be the address of the node to be deleted from a non-empty doubly linked list. Write the pseudocode to delete the node T.




***Algorithm DELETE_NONEMPTY_DLL(FIRST, LAST, T)***

1. $if\ T \rightarrow prev \neq NULL$
2. $\quad T \rightarrow prev \rightarrow next = T \rightarrow next$
3. $else$
4. $\quad FIRST = T \rightarrow next$
5. $\quad if\ FIRST \neq NULL$
6. $\quad\quad FIRST \rightarrow prev = NULL$
7. $if\ T \rightarrow next \neq NULL$
8. $\quad T \rightarrow next \rightarrow prev = T \rightarrow prev$
9. $else$
10. $\quad LAST = T \rightarrow prev$
11. $\quad if\ LAST \neq NULL$
12. $\quad\quad LAST \rightarrow next = NULL$
13. $RETNODE(T)$
14. $return$

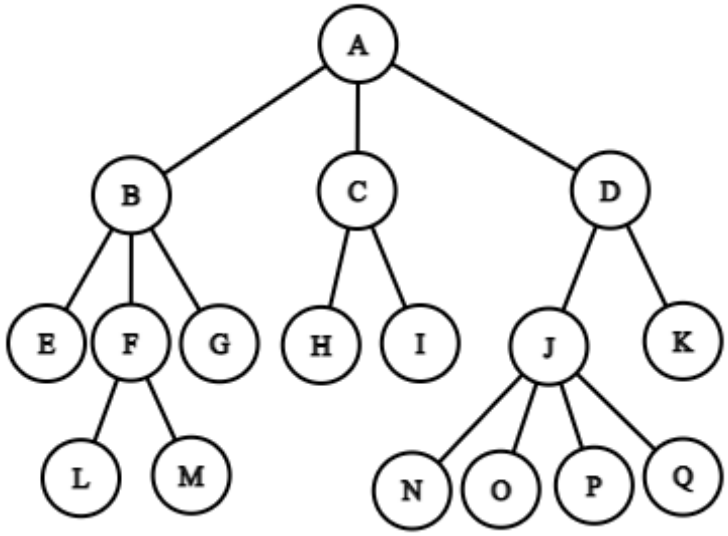
5. Write the parenthetical representation for the following general tree:

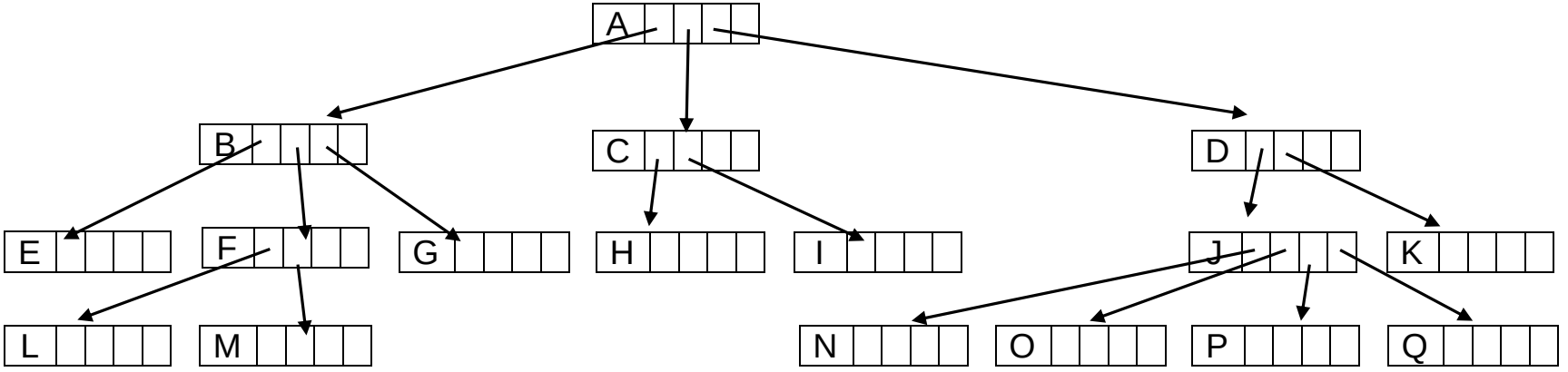



Parenthetical Representation:

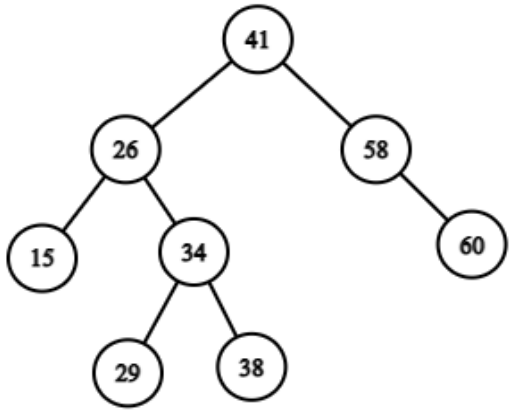
(A (B (E, F (L, M), G), C (H, I), D(J (N, O, P, Q) , K)))

6. Represent the following general tree using linked representation



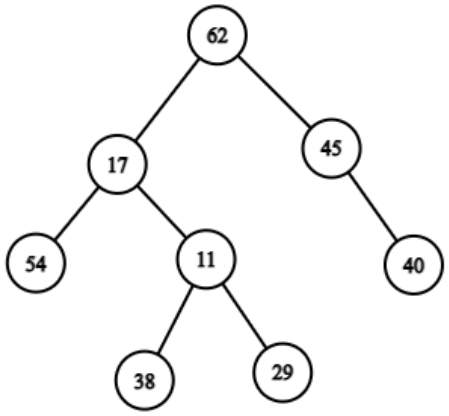



7. Represent the following binary tree as a sequential array.



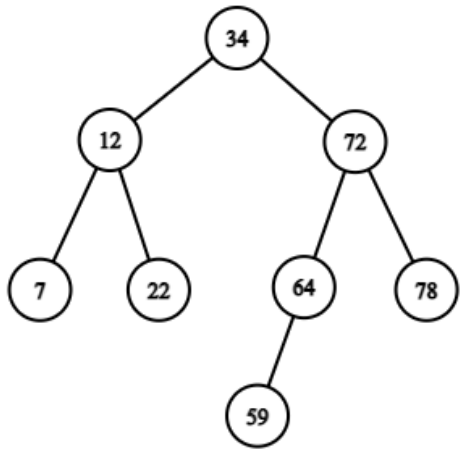

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 41 | 26 | 58 | 15 | 34 | - | 60 | - | - | 29 | 38 | - | - | - | - |

8. Find the inorder traversal for the following binary tree:




Inorder Traversal: 54, 17, 38, 11, 29, 62, 45, 40

9. Construct a binary search tree for the following input sequence:
34, 72, 12, 64, 59, 78, 22, 7



10. Write an algorithm to find minimum element in a binary search tree.

$\boldsymbol{Algorithm\ BST\_MINIMUM(ROOT)}$

1. $if\ ROOT = NULL$
2. $\quad Print$ Empty BST
3. $\quad return$
4. $T = ROOT$
5. $while\ T \rightarrow lchild \neq NULL$
6. $\quad T = T \rightarrow lchild$
7. $return\ T \rightarrow data$

## PART B

**Answer any THREE questions** **(3 x 10 = 30)**

11. Write the algorithm for adding two polynomials represented using singly linked list that store non-zero terms.

$\boldsymbol{Algorithm\ INSERT\_AT\_LAST(P, coef, exp)}$

$//\ To\ insert\ a\ new\ term\ of\ the\ polynomial\ at\ end$

1. $n = Allocate\_Node()$
2. $n \rightarrow coef = coef$
3. $n \rightarrow exp = exp$
4. $n \rightarrow link = NULL$
5. $if\ P.First = NULL$
6. $\quad n \rightarrow link = P.First$
7. $\quad P.First = P.Last = n$
8. $Else$
9. $\quad n \rightarrow link = P.Last \rightarrow link$
10. $\quad P.Last \rightarrow link = n$
11. $\quad P.Last = n$
12. $Return\ P$

$\boldsymbol{Algorithm\ ADD\_POLY(P, Q)}$

$//\ Adding\ two\ polynomials\ P\ and\ Q.$

1. $R.First = R.Last = NULL$
2. $t1 = P$
3. $t2 = Q$
4. $While\ t1\ != NULL\ and\ t2\ != NULL$
5. $\quad If\ t1 \rightarrow exp > t2 \rightarrow exp$
6. $\quad\quad R = INSERT\_AT\_LAST(R, t1 \rightarrow coef, t1 \rightarrow exp)$
7. $\quad\quad t1 = t1 \rightarrow link$
8. $\quad Else\ if\ t1 \rightarrow exp < t2 \rightarrow exp$
9. $\quad\quad R = INSERT\_AT\_LAST(R, t2 \rightarrow coef, t2 \rightarrow exp)$

10. $t2 = t2 \rightarrow link$
11. $Else$
12. $coef = t1 \rightarrow coef + t2 \rightarrow coef$
13. $exp = t1 \rightarrow exp$
14. $If\ coef\ != 0$
15. $R = INSERT\_AT\_LAST(R, coef, t2 \rightarrow exp)$
16. $t1 = t1 \rightarrow link$
17. $t2 = t2 \rightarrow link$
18. $While\ t1\ != NULL$
19. $R = INSERT\_AT\_LAST(R, t1 \rightarrow coef, t1 \rightarrow exp)$
20. $t1 = t1 \rightarrow link$
21. $While\ t2\ != NULL$
22. $R = INSERT\_AT\_LAST(R, t2 \rightarrow coef, t2 \rightarrow exp)$
23. $t2 = t2 \rightarrow link$
24. $Return\ R$

12. Write the algorithms to perform insertion, deletion, and search operations in an ordered singly linked list with head node.

***Algorithm INSERT_ORD_SLL(HEAD, x)***

***// Inserting an element x into an ordered singly linked list with head node***

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow link = NULL$
4. $prev = NULL$
5. $cur = HEAD$
6. $while\ cur \rightarrow link \neq NULL\ and\ cur \rightarrow link \rightarrow data \leq x$
7. $\quad prev = cur$
8. $\quad cur = cur \rightarrow link$
9. $T \rightarrow link = cur \rightarrow link$
10. $cur \rightarrow link = T$
11. $return$

***Algorithm SEARCH_ORD_SLL(HEAD, x)***

***//Searching for an element x in an ordered singly linked list with head node***

1. $cur = HEAD \rightarrow link$
2. $count = 1$
3. $while\ cur \neq NULL\ and\ cur \rightarrow data \neq x$
4. $\quad count = count + 1$
5. $\quad cur = cur \rightarrow link$
6. $if\ cur = NULL$

7. $print\ "Element\ not\ found"$
8. $else$
9. $print\ "Element\ found\ at\ position\ ", count$
10. $return$

***Algorithm DELETE_ORD_SLL(HEAD, x)***

***//Deleting an element x in an ordered singly linked list with head node***

1. $prev = HEAD$
2. $cur = HEAD \rightarrow link$
3. $while\ cur \neq NULL\ and\ cur \rightarrow data \neq x$
4. $prev = cur$
5. $cur = cur \rightarrow link$
6. $if\ cur = NULL$
7. $print\ "Element\ not\ found"$
8. $else$
9. $T = cur$
10. $prev \rightarrow link = cur \rightarrow link$
11. $RETNODE(T)$
12. $return$

13. Write the algorithms to perform insertion at beginning, insertion at end, insertion at specific location into a circular doubly linked list.

***Algorithm INSERT_AT_BEG_CDLL(FIRST, LAST, x)***

*// Inserting an element x at the begining of a circular doubly linked list*

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow prev = T \rightarrow next = NULL$
4. $if\ FIRST = NULL$
5. $T \rightarrow prev = T \rightarrow next = T$
6. $FIRST = LAST = T$
7. $else$
8. $T \rightarrow prev = LAST$
9. $T \rightarrow next = FIRST$
10. $LAST \rightarrow next = T$
11. $FIRST \rightarrow prev = T$
12. $FIRST = T$
13. $return$

***Algorithm INSERT_AT_END_CDLL(FIRST, LAST, x)***

*// Inserting an element x at the begining of a circular doubly linked list*

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow prev = T \rightarrow next = NULL$
4. $if\ FIRST = NULL$
5. $\quad T \rightarrow prev = T \rightarrow next = T$
6. $\quad FIRST = LAST = T$
7. $else$
8. $\quad T \rightarrow prev = LAST$
9. $\quad T \rightarrow next = FIRST$
10. $\quad LAST \rightarrow next = T$
11. $\quad FIRST \rightarrow prev = T$
12. $\quad LAST = T$
13. $return$

***Algorithm INSERT_AT_POS_CDLL(FIRST, LAST, x, p)***

*// Inserting an element x at given a position p of a circular doubly linked list*

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow prev = T \rightarrow next = NULL$
4. $if\ FIRST = NULL \qquad // Insert\ in\ empty\ CDLL$
5. $\quad T \rightarrow prev = T \rightarrow next = T$
6. $\quad FIRST = LAST = T$
7. $\quad return$
8. $if\ p = 1 \qquad // Insert\ as\ first\ node$
9. $\quad T \rightarrow prev = LAST$
10. $\quad T \rightarrow next = FIRST$
11. $\quad LAST \rightarrow next = T$
12. $\quad FIRST \rightarrow prev = T$
13. $\quad FIRST = T$
14. $\quad return$
15. $count = 1$
16. $cur = FIRST$
17. $while\ cur \rightarrow next \neq FIRST\ and\ count < p - 1$
18. $\quad count = count + 1$
19. $\quad cur = cur \rightarrow next$
20. $T \rightarrow prev = cur$
21. $T \rightarrow next = cur \rightarrow next$
22. $cur \rightarrow next \rightarrow prev = T$
23. $cur \rightarrow next = T$
24. $if\ LAST = cur$
25. $\quad LAST = T$
26. $return$

14. Write the algorithms to perform insertion, search, and preorder, inorder, and postorder traversals in a binary search tree

***Algorithm INSERT_BST(ROOT, x)***

*// To insert a new element x into a BST*

```
1.  T = GETNODE()
2.  T→data = x
3.  T→lchild = T→rchild = NULL
4.  if ROOT = NULL
5.      ROOT = T
6.      return
7.  parent = NULL
8.  cur = ROOT
9.  while cur ≠ NULL
10.     parent = cur
11.     if x < cur→data
12.         cur = cur→lchild
13.     else if x > cur→data
14.         cur = cur→rchild
15.     else
16.         Print "Duplicate value. Cannot insert"
17.         return
18. if x < parent→data
19.     parent→lchild = T
20. else
21.     parent→rchild = T
22.  return
```

***Algorithm SEARCH_BST(ROOT, x)***

*// To find an element x in the BST*

```
1.  if ROOT = NULL
2.      print "Empty Binary Search Tree."
3.      return
4.  cur = ROOT
5.  while cur ≠ NULL
6.      if x < cur→data
7.          cur = cur→lchild
8.      else if x > cur→data
9.          cur = cur→rchild
10.     else
```

11. $Print\ "Element\ Found"$
12. $return$
13. $Print\ "Element\ Not\ Found"$
14. $return\ root$

### $\boldsymbol{Algorithm\ INORDER(T)}$

$//\ To\ traverse\ the\ BST\ in\ inorder: Left, Data, Right$

1. $if\ T\ \neq\ NULL$
2. $\quad Inorder(T \rightarrow lchild)$
3. $\quad Print\ T \rightarrow data$
4. $\quad Inorder(T \rightarrow rchild)$

### $\boldsymbol{Algorithm\ PREORDER(T)}$

$//\ To\ traverse\ the\ BST\ in\ preorder: Data, Left, Right$

1. $if\ T\ \neq\ NULL$
2. $\quad Print\ T \rightarrow data$
3. $\quad Preorder(T \rightarrow lchild)$
4. $\quad Preorder(T \rightarrow rchild)$

### $\boldsymbol{Algorithm\ POSTORDER(T)}$

$//\ To\ traverse\ the\ BST\ in\ postorder: Left, Right, Data$

1. $if\ T\ \neq\ NULL$
2. $\quad Postorder(T \rightarrow lchild)$
3. $\quad Postorder(T \rightarrow rchild)$
4. $\quad Print\ T \rightarrow data$