CIRCULAR SINGLY LINKED LIST

1. Algorithm to insert an element into beginning of a circular singly linked list.

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

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow link = NULL$
4. $if\ FIRST = NULL$
5. $\quad T \rightarrow link = T$
6. $else$
7. $\quad temp = FIRST$
8. $\quad while\ temp \rightarrow link\ != FIRST$
9. $\quad\quad temp = temp \rightarrow link$
10. $\quad end\ while$
11. $\quad T \rightarrow link = FIRST$
12. $\quad temp \rightarrow link = T$
13. $endif$
14. $FIRST = T$
15. $return$

2. Algorithm to insert an element at the end of a circular singly linked list.

$\boldsymbol{Algorithm\ INSERT\_AT\_END\_CSLL(FIRST, x)}$

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow link = NULL$
4. $if\ FIRST = NULL$
5. $\quad T \rightarrow link = T$
6. $\quad FIRST = T$
7. $else$
8. $\quad temp = FIRST$
9. $\quad while\ temp \rightarrow link\ != FIRST$
10. $\quad\quad temp = temp \rightarrow link$
11. $\quad end\ while$
12. $\quad T \rightarrow link = temp \rightarrow link$
13. $\quad temp \rightarrow link = T$
14. $endif$
15. $return$

3. Algorithm to insert an element after a node with address P in a circular singly linked list.

$\boldsymbol{Algorithm\ INSERT\_AFTER\_CSLL(FIRST, P, x)}$

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

4. Algorithm to insert an element ina given position p, in a circular singly linked list.

$\boldsymbol{Algorithm\ INSERT\_ATPOS\_CSLL(FIRST, x, p)}$

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow link = NULL$
4. $if\ FIRST = NULL$
5. $\quad T \rightarrow link = T$
6. $\quad FIRST = T$
7. $else\ if\ p = 1$
8. $\quad T \rightarrow link = FIRST$
9. $\quad temp = FIRST$
10. $\quad while\ temp \rightarrow link\ != FIRST$
11. $\quad\quad temp = temp \rightarrow link$
12. $\quad end\ while$
13. $\quad temp \rightarrow link = T$
14. $\quad FIRST = T$
15. $else$
16. $\quad count = 1$
17. $\quad temp = FIRST$
18. $\quad while\ temp \rightarrow link\ != FIRST\ and\ count < p - 1$
19. $\quad\quad temp = temp \rightarrow link$

20. $count = count + 1$
21. $end\ while$
22. $T \rightarrow link = temp \rightarrow link$
23. $temp \rightarrow link = T$
24. $endif$
25. $return$

5. Algorithm to insert an element in an ordered circular singly linked list.

$\boldsymbol{Algorithm\ INSERT\_OCSLL(FIRST, x)}$

1. $T = GETNODE()$
2. $T \rightarrow data = x$
3. $T \rightarrow link = NULL$
4. $if\ FIRST = NULL$
5. $T \rightarrow link = T$
6. $FIRST = T$
7. $else\ if\ FIRST \rightarrow data \geq x$
8. $T \rightarrow link = FIRST$
9. $temp = FIRST$
10. $while\ temp \rightarrow link\ != FIRST$
11. $temp = temp \rightarrow link$
12. $end\ while$
13. $temp \rightarrow link = T$
14. $FIRST = T$
15. $else$
16. $temp = FIRST$
17. $while\ temp \rightarrow link\ != FIRST\ \&\&\ temp \rightarrow link \rightarrow data < x$
18. $temp = temp \rightarrow link$
19. $end\ while$
20. $T \rightarrow link = temp \rightarrow link$
21. $temp \rightarrow link = T$
22. $endif$
23. $return$

6. Algorithm to delete an element x from an ordered circular singly linked list.

$\boldsymbol{Algorithm\ DELETE\_OCSLL(FIRST, x)}$

1. $if\ FIRST = NULL\ or\ FIRST \rightarrow data > x$
2. $print\ "Element\ not\ present\ in\ the\ list"$

3. $return$
4. $end\ if$
5. $if\ FIRST \rightarrow data = x$
6. $T = FIRST$
7. $if\ FIRST \rightarrow link = FIRST$
8. $FIRST = NULL$
9. $else$
10. $FIRST = FIRST \rightarrow link$
11. $temp = FIRST$
12. $while\ temp \rightarrow link\ != T$
13. $temp = temp \rightarrow link$
14. $end\ while$
15. $temp \rightarrow link = FIRST$
16. $else$
17. $prev = FIRST$
18. $while\ prev \rightarrow link\ != FIRST\ \&\&\ prev \rightarrow link \rightarrow data < x$
19. $prev = prev \rightarrow link$
20. $end\ while$
21. $T = prev \rightarrow link$
22. $if\ T \rightarrow data = x$
23. $prev \rightarrow link = T \rightarrow link$
24. $else$
25. $print\ "Element\ not\ present\ in\ the\ list"$
26. $return$
27. $end\ if$
28. $endif$
29. $RETNODE(T)$
30. $return$

7. Algorithm to search for the position of an element x in an ordered circular singly linked list. Return -1 if element not present.

Algorithm SEARCH_OCSLL(FIRST, x)

1. $if\ FIRST = NULL\ or\ FIRST \rightarrow data > x$
2. $return - 1$
3. $end\ if$
4. $p = 1$
5. $T = FIRST$
6. $while\ T \rightarrow link\ != FIRST\ and\ T \rightarrow data < x$

7. $p = p + 1$
8. $T = T \rightarrow link$
9. $end\ while$
10. $if\ T \rightarrow data = x$
11. $return\ p$
12. $end\ if$
13. $return - 1$

CIRCULAR DOUBLY LINKED LIST

8. Algorithm to Insert an element x at the begining of a circular doubly linked list

Algorithm INSERT_AT_BEG_CDLL(FIRST, LAST, x)

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. $return$

9. Algorithm to Insert an element x at the end of a circular doubly linked list

Algorithm INSERT_AT_END_CDLL(FIRST, LAST, x)

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. $LAST = T$
13. $return$

10. Algorithm to insert an element x at given a position p of a circular doubly linked list

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

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. $return$
8. $if\ p = 1$
9. $T \rightarrow prev = LAST$
10. $T \rightarrow next = FIRST$
11. $LAST \rightarrow next = T$
12. $FIRST \rightarrow prev = T$
13. $FIRST = T$
14. $return$
15. $count = 1$
16. $cur = FIRST$
17. $while\ cur \rightarrow next \neq FIRST\ and\ count < p - 1$
18. $count = count + 1$
19. $cur = cur \rightarrow next$
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. $LAST = T$
26. $return$

11. Algorithm to insert into an ordered circular doubly linked list

Algorithm INSERT_OCDLL(FIRST, LAST, x)

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. $\quad return$
8. $endif$
9. $if\ FIRST \rightarrow data > x$
10. $\quad T \rightarrow prev = LAST$
11. $\quad T \rightarrow next = FIRST$
12. $\quad LAST \rightarrow next = T$
13. $\quad FIRST \rightarrow prev = T$
14. $\quad FIRST = T$
15. $\quad return$
16. $end\ if$
17. $cur = FIRST$
18. $while\ cur \rightarrow next \neq FIRST\ and\ cur \rightarrow next \rightarrow data < x$
19. $\quad cur = cur \rightarrow next$
20. $end\ while$
21. $T \rightarrow prev = cur$
22. $T \rightarrow next = cur \rightarrow next$
23. $cur \rightarrow next \rightarrow prev = T$
24. $cur \rightarrow next = T$
25. $if\ LAST = cur$
26. $\quad LAST = T$
27. $end\ if$
28. $return$

12. Algorithm to delete an element x from an ordered circular doubly linked list.

Algorithm DELETE_OCDLL(FIRST, LAST, x)

1. $if\ FIRST = NULL\ \ or\ FIRST \rightarrow data > x$
2. $\quad print\ "Element\ not\ present\ in\ list"$

3. $return$
4. $end\ if$
5. $cur = FIRST$
6. $while\ cur \rightarrow next \neq FIRST\ and\ cur \rightarrow data < x$
7. $cur = cur \rightarrow next$
8. $end\ while$
9. $if\ cur \rightarrow data = x$
10. $if\ FIRST = LAST = cur$
11. $FIRST = LAST = NULL$
12. $else$
13. $cur \rightarrow prev \rightarrow next = cur \rightarrow next$
14. $cur \rightarrow next \rightarrow prev = cur \rightarrow prev$
15. $if\ FIRST = cur$
16. $FIRST = FIRST \rightarrow next$
17. $else\ if\ LAST = cur$
18. $LAST = LAST \rightarrow prev$
19. $end\ if$
20. $end\ if$
21. $RETNODE(cur)$
22. $else$
23. $print\ "Element\ not\ present\ in\ list"$
24. $end\ if$
25. $return$

13. Algorithm to search for the position of an element x in an ordered circular doubly linked list. Return -1 if element not present.

***Algorithm SEARCH_OCDLL*($FIRST, LAST, x$)**

1. $if\ FIRST = NULL\ \ or\ FIRST \rightarrow data > x$
2. $return - 1$
3. $end\ if$
4. $cur = FIRST$
5. $p = 1$
6. $while\ cur \rightarrow next \neq FIRST\ and\ cur \rightarrow data < x$
7. $p = p + 1$
8. $cur = cur \rightarrow next$
9. $end\ while$
10. $if\ cur \rightarrow data = x$

11. $return\ p$
12. $end\ if$
13. $return -1$