Circular Linked List

Create Circular Linked List

- Creating a linked list

Algorithm create_linked_list(struct link *header)

1. [initialize]
    1. header->next= NULL
    2. Point the node to the header
2. Repeat while(ch!='n')
    1. node->next= create a dynamic list of type struct link
    2. node=node->next
    3. Input the data for the node
    4. node->next = null
    5. node_counts=node_counts +1
    6. Ask the user for continuing to create the list

Create Circular Linked List cont…

3. if(ch=='n')
   1. node->next=header
   2. node=node->next
   3. node->info=node_counts
4. Display " NO OF NODES IN THE LIST”

Displaying a linked list

Algorithm display_linked_list(struct link *header)

1. node=point to the first node
2. Repeat while(node!=header)
    1. Display node->info
    2. node=node-> next
3. if(node==header)

    display “TOTAL NO. OF NODES:”

Algorithm to Insert node in the middle of the list

Algorithm insert_in_the_middle(struct link *header)

1. [initialize]

    node_num=1;

    node=point to the next node

    previous=address of the header

2. Input the location at which the node is to be inserted.

Algorithm to Insert node in the middle of the list Continued..

3. if(insert_node<=node_counts)
   1. Repeat 1
      while(node!=header)
        if(node_num= insert_node)
            1. new1 =create a dynamic list of type struct link
            2. point new1 to node
            3. point previous to new1
            4. Enter the value for the new node new1
        else
            1. point node to the next node
            2. point previous to the next node
        node_num++
   2. node=header
      node-nfo=node_counts
   else
      display " THERE ARE ONLY 'node_counts' NODES IN THE LIST"

Algorithm to delete desired node from the list

Algorithm
delete_desired_node(struct link *header)

1.[initialize]
node_number=1
delete_node=0
node=point to the first node
previous=address of the header

2. Enter the node number you want to delete

3. Repeat while(node!=header)
1.if node_number=delete_node
1. previous->next= node->next
2. free(node)

else
1. node=point node to the next node
2. previous=point previous to the next node
node_number++