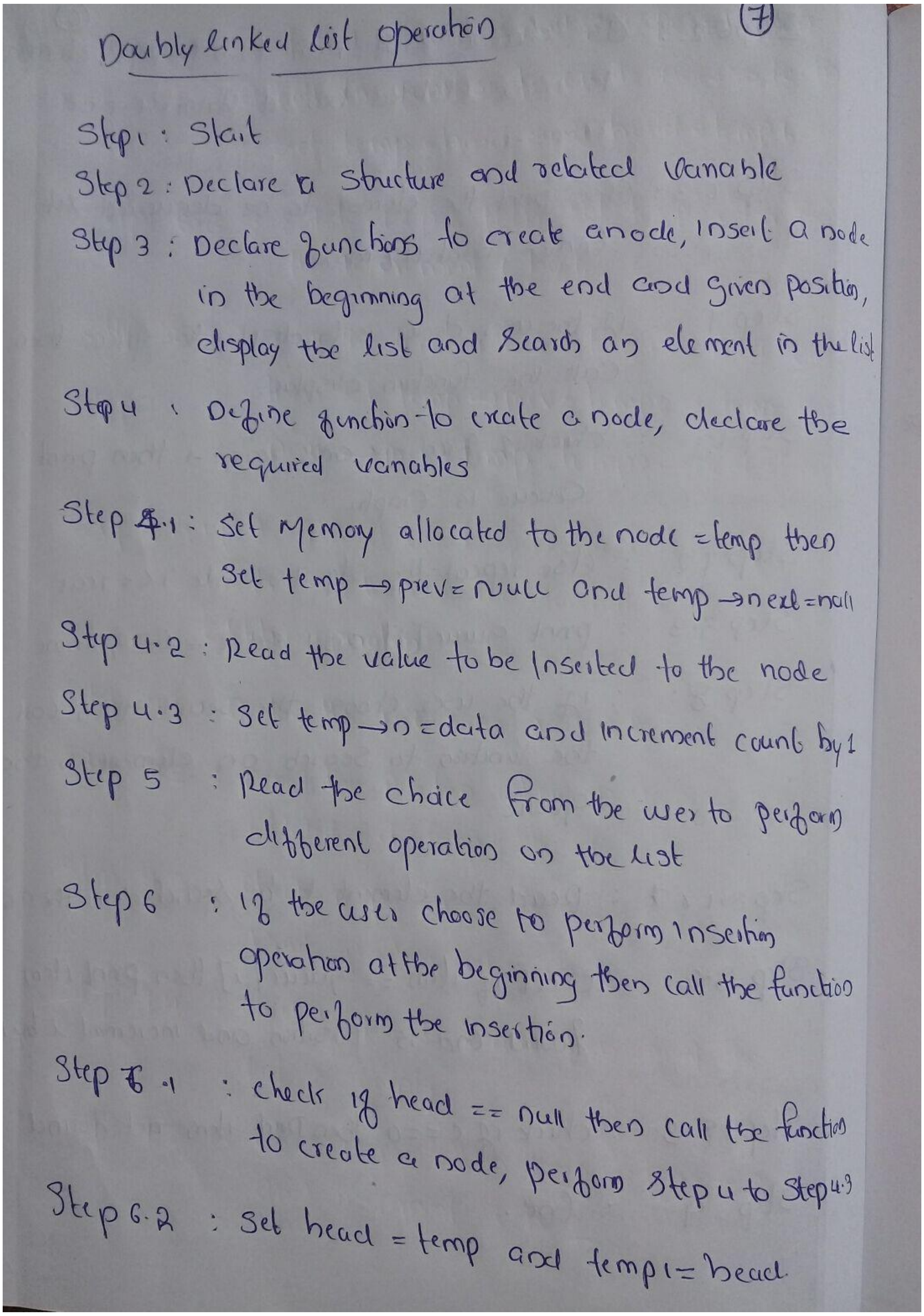
ALGORITHM:



Doubly linked list operation

```
Step1 : Start
Step 2 : Declare a structure and related variable
Step 3 : Declare functions to create a node, insert a node
         in the beginning at the end and given position,
         display the list and search an element in the list
Step 4 : Define function to create a node, declare the
         required variables
Step 4.1 : Set memory allocated to the node = temp then
           set temp → prev = null and temp → next = null
Step 4.2 : Read the value to be inserted to the node
Step 4.3 : Set temp → n = data and increment count by 1
Step 5   : Read the choice from the user to perform
           different operation on the list
Step 6   : If the user choose to perform insertion
           operation at the beginning then call the function
           to perform the insertion.
Step 6.1 : check if head == null then call the function
           to create a node, perform step 4 to step 4.3
Step 6.2 : Set head = temp and temp1 = head
```

Step 6.3 : Else call the function to create a node. Perform step 4 to 4.3 then set temp→next = head. Set head→prev = temp and head = temp

Step 7 : If the user choice is to perform insertion at the end of the list, then call the function to perform the insertion at the end.

Step 7.1 : check if head == null then call the function to create a newnode then set temp = head and set head = temp1

Step 7.2 : Else call the function to create a new node then set temp1→next = temp, temp→prev = temp1 and temp1 = temp.

Step 8 : If the user choose to perform insertion in the list at any position then call the function to perform the insertion operation.

Step 8.1 : Declare the neccassary variable.

Step 8.2 : Read the position where the node and to the inserted, set temp2 = head.

Step 8.3 : Check if pos < 1 or pos >= count +1 then print the position is out of range.

Step 8.4 : check if head == null and pos = 1 then print "empty list cannot insert other 1st posi"

Step 8.5 : check if head ==null and pos =1 then call the function to create newNode. then Set temp =head and head =temp1.

Step 8.6 : while i<pos then set temp2 = temp2→next the increment i by 1.

Step 8.7 : call the function to create a newNode and then set temp→prev = temp2 , temp→next=temp2 next→prev = temp, temp2→next =temp

Step 9 : If the user choose to perform deletion operation is the list then all the function to perform the deletion operation.

Step 9.1 : Declare the necessary variables

Step 9.2 : Read the position where node need to be deleted set temp 2=head.

Step 9.3 : Check if pos <1 or pos >= count+1. then Print position out of range

Step 9.4 : check if head == null then print the list is empty.

Step 9.5 : while i< pos then temp2 =temp2→next and increment i by 1

Step 9.6 : check if c=1 then check if temp2 → next == null then print node deleted free (temp2) set temp2 = head = null.

Step 9.7 : check if temp2 → next == null then temp2 → prev → next = null then free (temp2) then print node deleted.

Step 9.8 : temp2 → next → prev = temp2 → prev then check if i != 1 then temp2 → prev → next = temp2 → next.

Step 10 : if the user choose to perform the display operation then call the function to display the list.

Step 10.1 : set temp2 = n.

Step 10.2 : check if temp2 = null then print list is empty

Step 10.3 : while temp2 → next != null then print temp2 → n then temp2 = temp2 → next

Step 11 : if the user choose to perform the search operation then call the function to perform search operations

Step 11.1 : Declare the necessary variables

Step 11.2 : Set temp2 = head.

Step 11.3 : check if temp2 == null then print the list is empty.

Step 11.4 : Read the value to be searched.

Step 11.5 : while ~~st~~ temp2 != null the check if temp2→n = = data
then print element found at position count+1

Step 11.6 : Else set temp2 = temp2 → next and increment
count by 1.

Step 11.7 : print element not found in the list

Step 12 : End.