LAB ASSIGNMENT - 7

Ques: Write a menu driven programe to implement single link list must include following :-

1. Create
2. Insert at front
3. Insert at end
4. insert before a node
5. insert after a node
6. delete at front
7. delete at end.
8. delete before a node
9. delete after a node
10. Search
11. Display

Sol- Source Code -

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <Malloc.h>
```

DATE: / / Aa
PAGE NO.

```
struct node
{
  int data;
  struct node *next;
} *start = NULL, *q, *t, *temp, *prevode.

int main()
{
  int ch.;
  Void insert_beg();
  Void insert_end()
  Void Display()
  Void delete_beg()
  Void delete_end()
  struct node *create_ll(struct node *);
while(ch!=1)
{
  Printf("******* MAIN MENU ******* ");
  Printf("1 Create a list\n");
  Printf("2 insert at front\n");
  Printf("3 insert at end\n");
  Printf("4 insert before a node\n");
  Printf("5 insert after a node\n");
  Printf("6 deleted at front\n");
  Printf("7 delete at end\n");
  Printf("8 delete after a node\n");
  Printf("9 delete before a node\n");
  Printf("10 Search\n");
  Printf("11 Display\n");
  Printf("12 EXIT\n");
  Printf("Enter your choice (1-12)");
  Scanf("%od", &ch);
```

```
switch (ch)
{
    Case 1 : start = create_ll (start);
             break;
    Case 2 : insert_beg ();
             break;
    Case 3 : insert_end ();
             break;
    Case 4 : insert_before ();
             break;
    Case 5 : insert_after ();
             break;
    Case 6 : delete_beg ();
             break;
    Case 7 : delete_end ();
             break;
    Case 8 : delete_before ();
             break;
    Case 9 : delete_after ();
             break;
    Case 10 : Search ()
              break;
    Case 11 : display ();
              break;
    Case 12 : exit (0);
              break;
    default : printf ("Wrong choice!");
  }
}
return 0;
}
```

Always ahead

```
Void insert_beg ()
{
  int num;
  t = (struct node *) malloc (size of (struct
                                node)).
  Printf ("Enter data");
  Scanf ("%d", &num);
  t -> data = num;

  if (start == NULL)
  {
     t -> next == NULL;
     start = t;
  }
  else
  {
     t -> next = start;
     start = t;
  }
}

Void insert_end ()
{
  int num;
  t = (Struct node *) malloc (size of (struct node));
  Printf ("Enter Data");
  Scanf ("%d", &num);
  t -> data = num;
  t -> next = NULL;

  if (start == NULL)
```

```
    }
        start = t;
    }
    else
    {
        q = start;
        while (q -> next != NULL)
        q = q -> next;
        q -> next = t;
    }
}
Void display ()
{
    if (start == NULL)
    {
        Printf ("List is empty");
    }
    else
    {
        q = start;
        printf ("The linked list is \n");
        while (q != NULL)
        {
            printf ("%d->", q -> data);
            q = q -> next;
        }
    }
}
Void delete beg ()
{
    if (start == NULL)
    {
```

```
        printf("The list is empty!");
    }
    else
    {
        q = start;
        start = start -> next;
        printf("Deleted element is %d", q->data);
        free(q);
    }
}
void delete_end()
{
    if(start == NULL)
    {
        Printf("The list is empty!!");
    }
    else
    {
        q = start;
        while(q->next->next != NULL)
        q = q->next;
        t = q->next;
        q->next = NULL;
        Printf("Deleted elemend is %d", t->data);
        free(t);
    }
}
Struct node *Create_ll(struct node *start)
{
Struct node *newnode, *ptr;
```

```
int val;
printf("enter the new value");
Scanf("%od", &val);
new_node = (struct node *) malloc (sizeof struct node));
new_node -> data = val;
new_node -> next = NULL;
ptr = start;
    if (start == NULL)
    {
        start = new_node;
    }
    else
    {
      while (ptr -> next != NULL)
    {
      Ptr = ptr -> next;
    }
    Ptr -> next = new_node;
  }
return start;
}
  Void insert_before ( )
   {
    printf(" Enter the position where you want to
    insert a new node \n");
    Scanf ("%od", &pos)
    temp = start;
    while ( temp != NULL)
    {
       temp = temp -> next;
       Count ++;
    }
```

Always ahead

```
if (pos > count)
  printf ("Invalid position \n");
else {
    else
    {
      temp = start;
      while (i< pos-1)
      {
        temp= tem -> next;
        i++;
      }
    struct node * link = (struct node * malloc(sizeof
                (struct node));
    printf("Enter data \n");
    Scanf("%d", &link -> data);
    link -> next = temp -> next;
    temp -> next = link;
}
}
  Void insert_after()
{
  printf("Enter the position where you want
   to insert a new node \n");
  Scanf("%d",  &pos
  temp = start;
  while (temp != NULL)
  {
     temp = temp -> next.
     Count ++;
  }
if (pos > count)
printf ("Invalid position \n");
```

Always ahead

```
{
else
{
    temp = start
    while (i<pos)
    {
        temp = temp -> next ;
    i++
    }
Structure node *link = (struct node *)
                       malloc (size of (struct
                       node));

printf (" Enter data\n");
Scanf ("%d", &link -> data);
link -> next = temp -> next;
  temp -> next = link ;
}
}

Void delete_before ()
{
  temp = start;
  printf (" Enter position \n");
  Scanf("%od", &.pos);

  While ( i< pos -1)
  {
     premode = temp
     temp = tem -> next
     i++ ;
  }
```

```
    prevenode -> next = temp -> next;
    free (temp)
}
Void delete_after()
{
    temp = short
    printf (" Enter position \n");
    Scanf ("%d", &pos);

    while (i < pos)
    {
     temp = temp -> next;
        i++;
    }

    nextnode = temp -> next;
    temp -> next = nextnode -> next.;
    free (nextnode)
}


Void Search ()
    Pos = 1;
    printf (" Enter the data to search \n")
    Scanf("%d", &item);
    temp = start
    while (temp != NULL)
     {
         if (temp -> data == item)
         {
           Printf ("Item found at %d position
                                    \n") pos );

         return
```

Always ahead..

```
    }
    temp = temp -> next;
    pos ++;
  }else{
    printf("Item not found \n");
  }
```

OUTPUT

Enter 1 Create a list.
Enter 2 insert at beg.
Enter 3 insert at end
Enter 4 insert before a node
Enter 5 insert after a node
Enter 6 delete at beg.
Enter 7 delete at end
Enter 8 delete before a node
Enter 9 delete a node after a node
Enter 10 Search
Enter 11 - Display
Enter 12 - Exit

Algorithem

Step 1 START

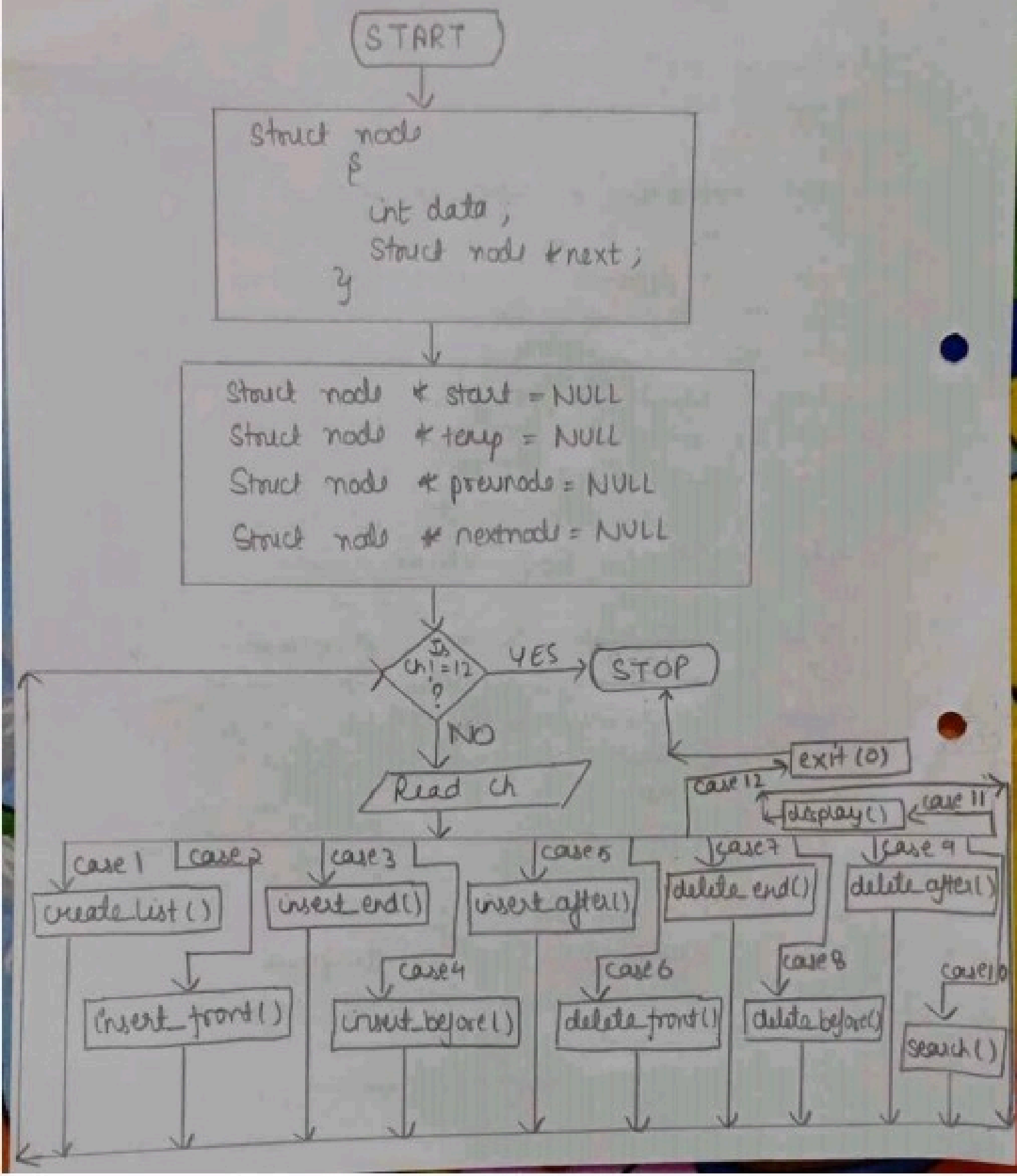
Step 2 Declare a nod structure member data and next
Step 3 Declare a node pointer.
Step 4 Take choice inch
Step 5 for Case 1 Call Createlist() and so on

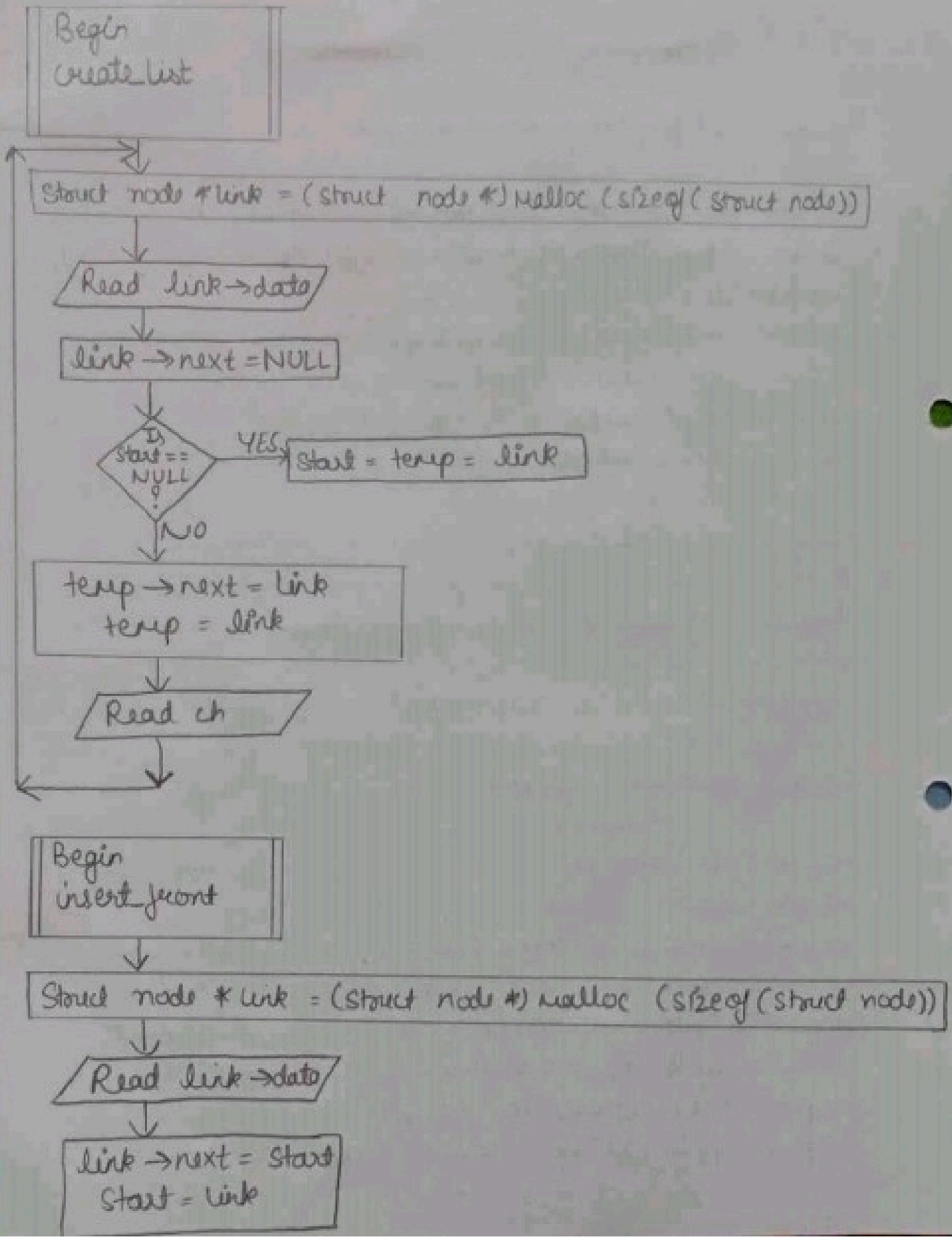
Always ahead

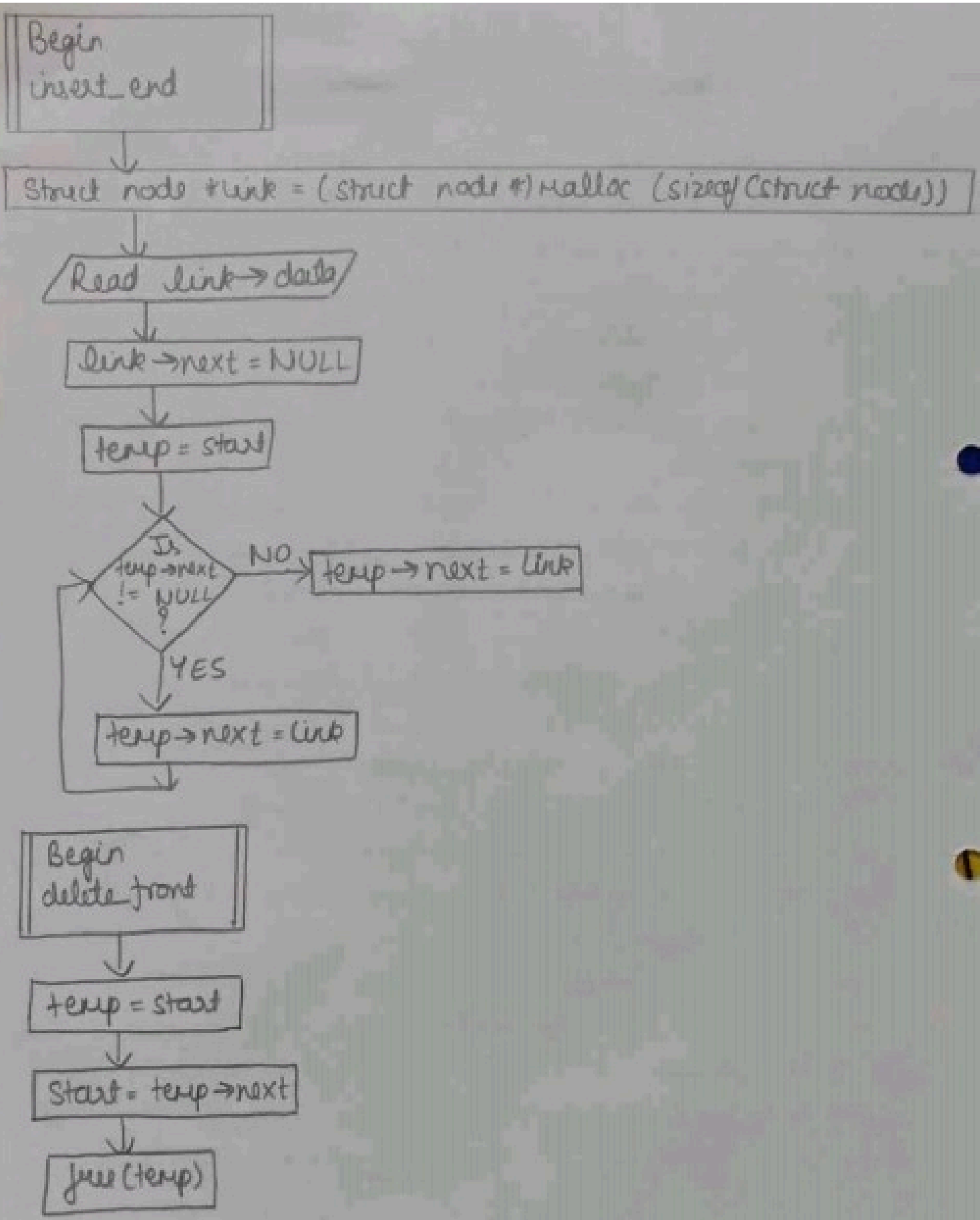
Step 6 Create_list(), insert_front(), insert_end(), insert_before(), insert_after(), delete_front(), delete_end(), delete_before(), delete_after(), Search(), display(),

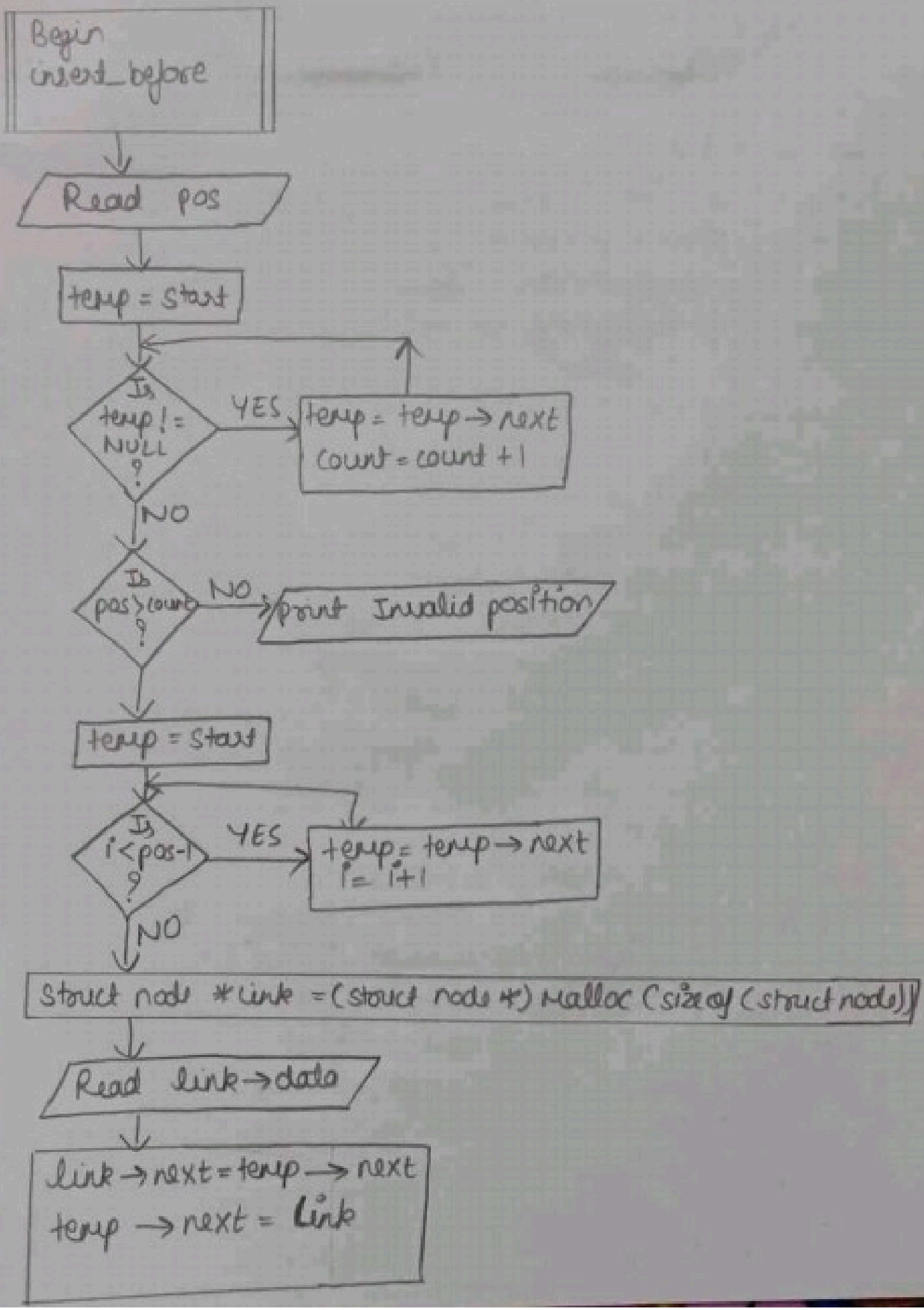
Step 7 STOP.

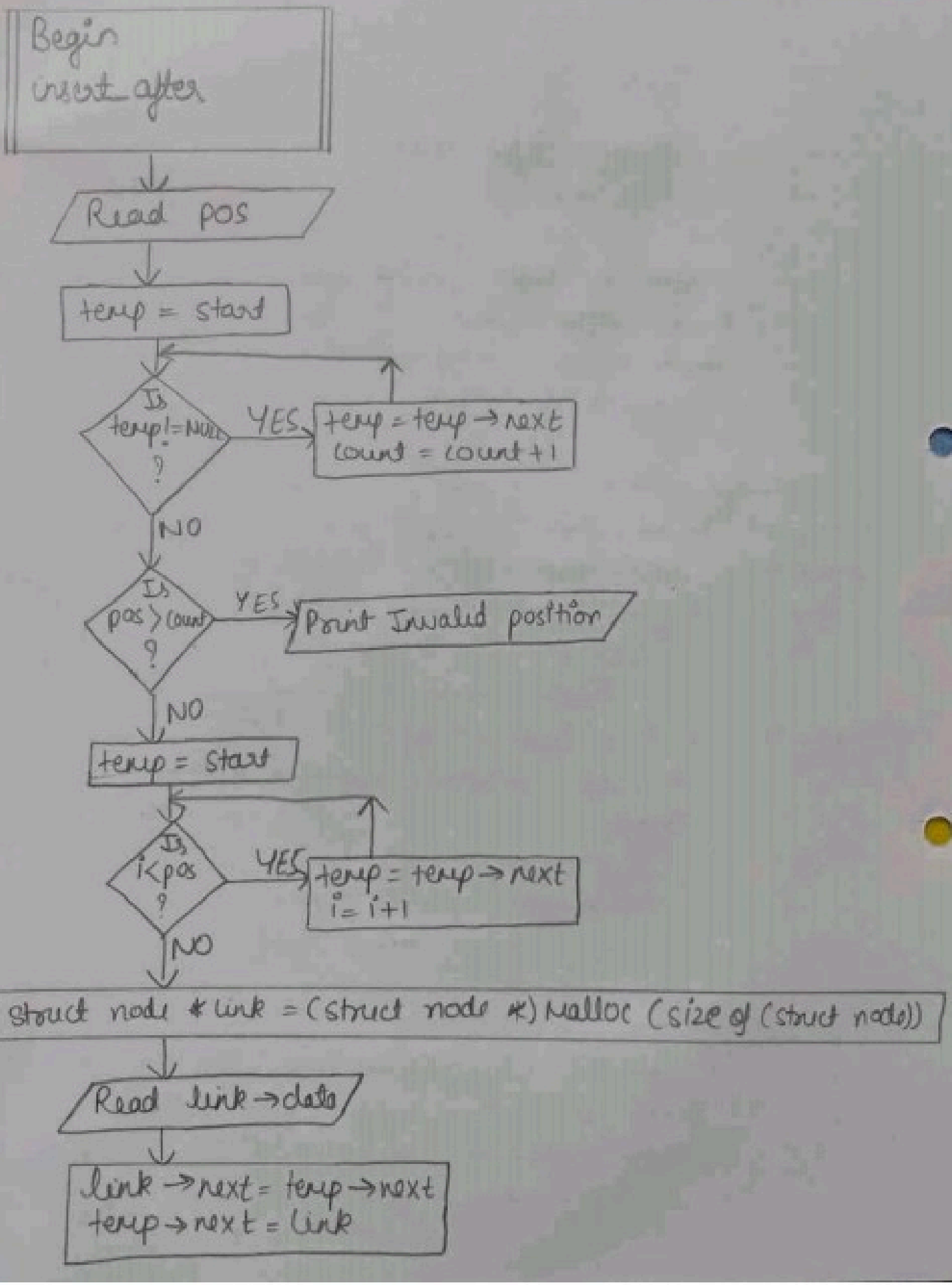
FLOWCHART

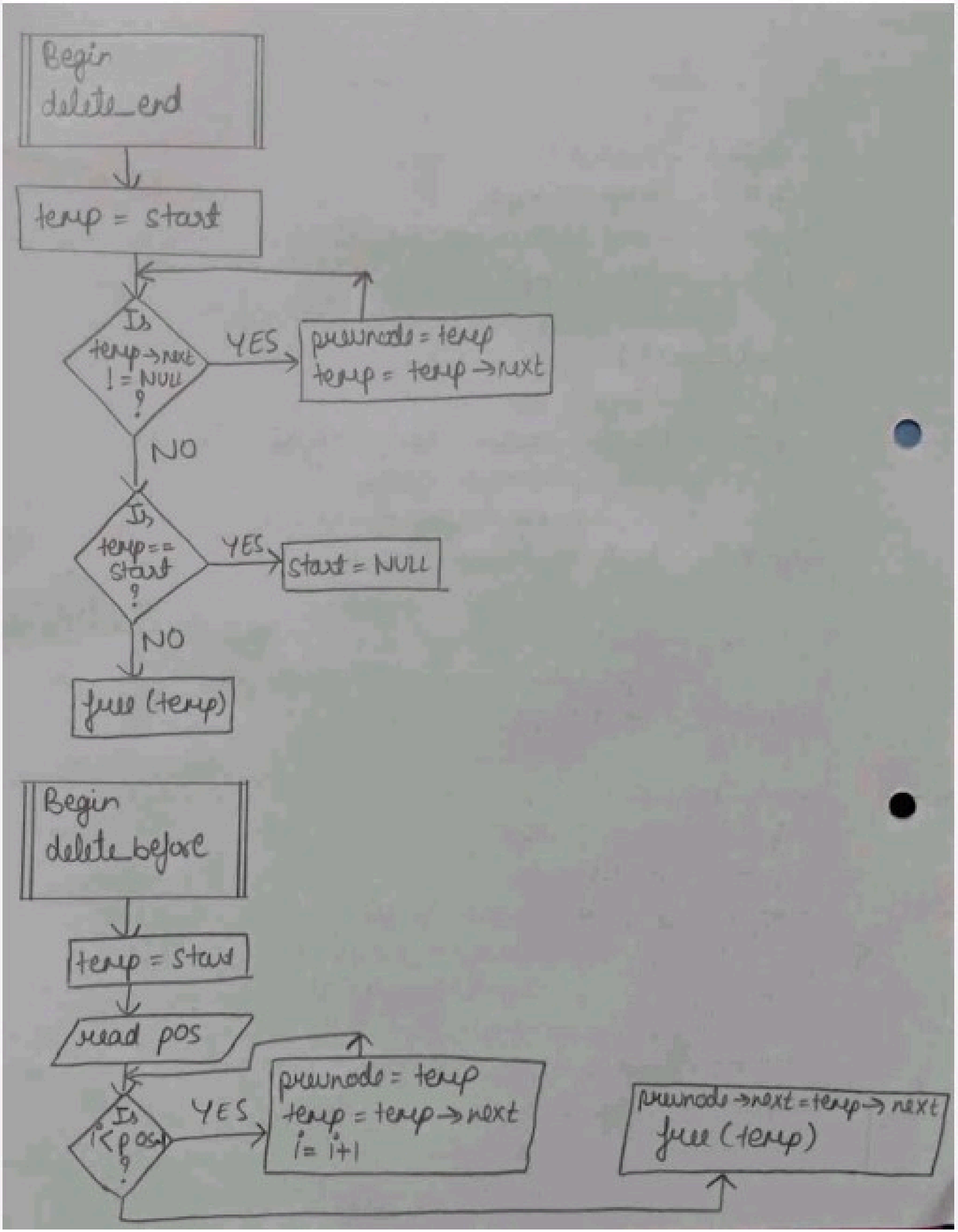

START
Struct node
{
int data;
Struct node *next;
};
Struct node * start = NULL
Struct node * temp = NULL
Struct node * prevnode = NULL
Struct node * nextnode = NULL
Is ch!=12 ?
YES
STOP
NO
Read ch
exit (0)
case 12
display()
case 11
case 1
case 2
case 3
case 5
case 7
case 9
create_list()
insert_end()
insert_after()
delete_end()
delete_after()
insert_front()
case 4
insert_before()
case 6
delete_front()
case 8
delete_before()
case 10
search()

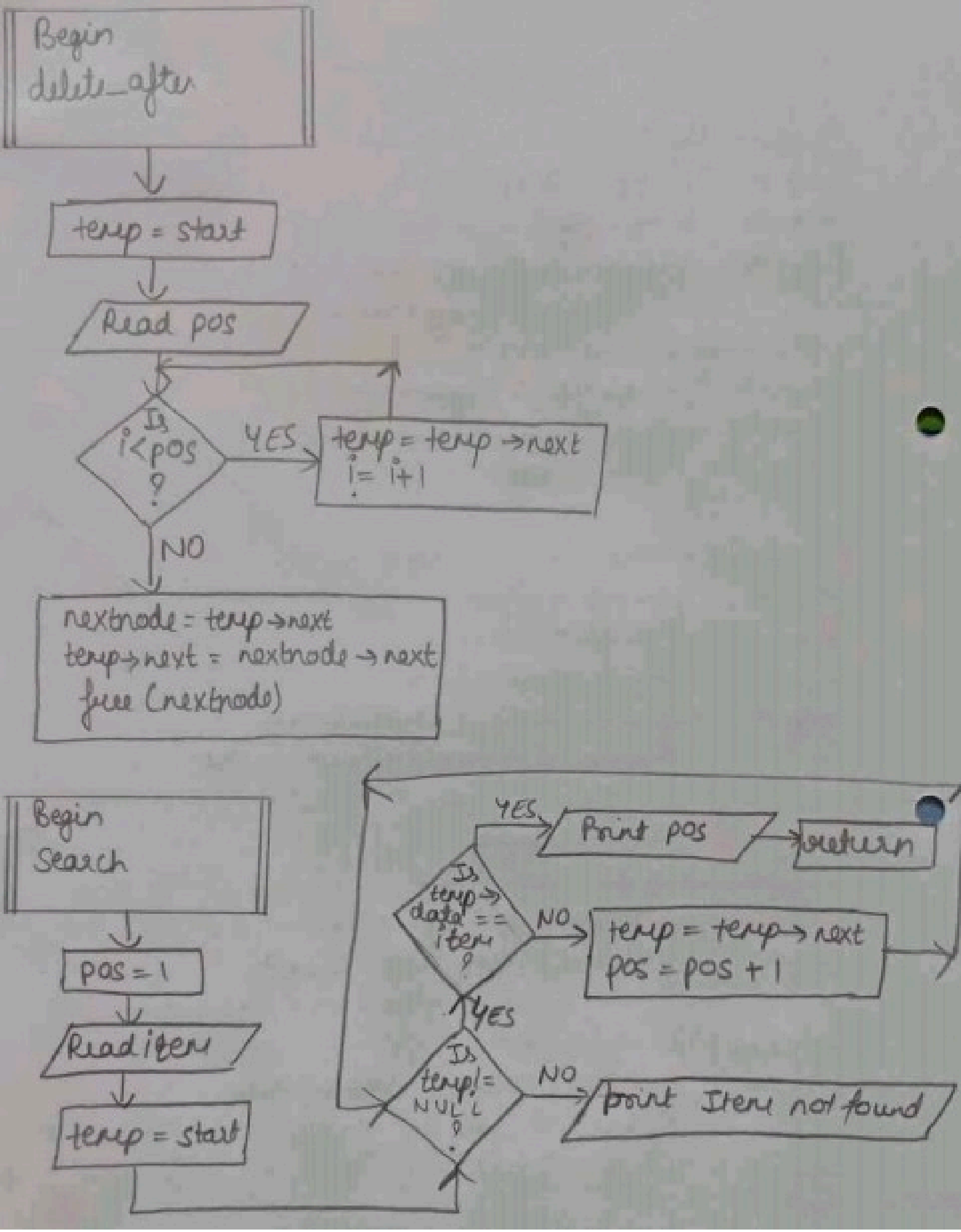

Begin
create_list
Struct node *link = (struct node *) malloc (sizeof(struct node))
Read link->data
link->next =NULL
Is start == NULL ?
YES
start = temp = link
NO
temp->next = link
temp = link
Read ch
Begin
insert_fcont
Struct node *link = (struct node *) malloc (sizeof(struct node))
Read link->data
link->next = start
start = link


Begin
insert_end
Struct node *link = (struct node *) malloc (sizeof(struct node))
Read link→data
link→next = NULL
temp = start
Is temp→next != NULL ?
NO
temp→next = link
YES
temp→next = link
Begin
delete_front
temp = start
start = temp→next
free(temp)


Begin insert_before
Read pos
temp = start
Is temp != NULL ?
YES
temp = temp -> next
count = count + 1
NO
Is pos > count ?
NO
Print Invalid position
temp = start
Is i < pos-1 ?
YES
temp = temp -> next
i = i+1
NO
Struct node *link = (struct node *) malloc (sizeof (struct node))
Read link -> data
link -> next = temp -> next
temp -> next = link


Begin
insert after
Read pos
temp = start
Is temp != NULL ?
YES
temp = temp → next
count = count + 1
NO
Is pos > count ?
YES
Print Invalid position
NO
temp = start
Is i < pos ?
YES
temp = temp → next
i = i + 1
NO
struct node * link = (struct node *) malloc (size of (struct node))
Read link → data
link → next = temp → next
temp → next = link


Begin
delete_end
temp = start
Is temp->next != NULL ?
YES
prevnode = temp
temp = temp->next
NO
Is temp == start ?
YES
start = NULL
NO
free (temp)
Begin
delete_before
temp = start
read pos
Is i<pos ?
YES
prevnode = temp
temp = temp->next
i = i+1
prevnode->next = temp->next
free (temp)


Begin
delete_after
temp = start
Read pos
Is i<pos ?
YES
temp = temp → next
i = i+1
NO
nextnode = temp → next
temp → next = nextnode → next
free (nextnode)
Begin
Search
pos = 1
Read item
temp = start
Is temp → data == item ?
YES
Print pos
return
NO
temp = temp → next
pos = pos + 1
Is temp != NULL ?
YES
NO
Print Item not found