CSC303 Data Structure

University Questions

1. Explain types of data structure with example

2. What is non-linear data structure? Explain with example.

3. Explain the concept of ADT and describe Queue ADT.

4. Define Data Structure. Differentiate linear and non-linear data structures with example.

5. Write a C program to implement stack using array.

ANSWER:

```
#include<stdio.h>
#define STACKSIZE 5

struct bufferstack
{
int stk[STACKSIZE];
int top; // We will use it as POINTER to top of the stack
}s; // Here s is struct variable, you can see here how to implement structure in C

void push(); // To push elements in stack
int  pop();  // To Pop elements in stack
void display();

int main()
{
int c;
s.top=-1; //Set s to -1
int x=1;
```

```
while(x)  //While loop to keep program in loop
  {
printf("\n TOP is at : %d\t",s.top+1); //this line is to test, the position of s
printf("\n ****MENU****\n");
printf("1: Push \n");
printf("2: Pop \n");
printf("3: Display \n");
printf("4: Exit");
printf("\n Please enter your choice : ");
scanf("%d",&c);
    switch(c)
    {

    case 1:
        push();
        break;
    case 2:
        pop();
        break;
    case 3:
        display();
        break;
    case 4:
        return 0;
    }
printf(" \n Do you want to cotin....? press 1 or 0 ");
scanf("%d",&x);
```

```
  }
}

void push()
{
   int num;
   printf("\n ***PUSH OPERATION***");
   if(s.top==(STACKSIZE-1))
   {
      printf("\n Sorry You can't push any element into stack .... ,Stack is full");
   }
   else
   {
   printf("\n Enter the number to push into stack:-\t");
   scanf("%d",&num);
   s.top+=1;
   s.stk[s.top]= num;

   }
}

int pop()
{
   printf("\n ***POP OPERATION***");
   int num;
   if(s.top==-1)
   {
```

```
   printf("\nstack is empty ");
   }
   else
   {
   num=s.top;
   printf("\n Poped number is : %d\t",s.stk[num]);
   s.stk[num]=0; //To delete top element from stack
   s.top-=1;
   }
   return num;

}

void display()
{
   int i;
   printf("\n ***DISPLAY OPERATION***");
   if(s.top==-1)
   {
   printf("\n Stack is empty \n");
   }
   else
   {
   for(i=s.top;i>=0;i--)
      {
      printf("\n%d : %d",i,s.stk[i]);
```

```
        }
    printf("\tTOP=%d",s.top);
    }
}
```

6. Write a program in C to implement stack using linked list. Perform the following operations

i. Push

ii. Pop

iii. Peek

iv. Display the stack contents

ANSWER:

```
/*
 * C Program to Implement a Stack using Linked List
 */
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int info;
    struct node *ptr;
}*top,*top1,*temp;



void push(int data);
void pop();
int peek();
void display();
void create();

int count = 0;

void main()
{
   int no, ch, e;
          top = NULL;

   printf("\n 1 - Push");
   printf("\n 2 - Pop");
   printf("\n 3 - Peek");
   printf("\n 4 - Empty");
   printf("\n 5 - Exit");

   while (1)
   {
      printf("\n Enter choice : ");
      scanf("%d", &ch);

      switch (ch)
      {
      case 1:
         printf("Enter data : ");
```

```
        scanf("%d", &no);
        push(no);
        break;
    case 2:
        pop();
        break;
    case 3:
        if (top == NULL)
            printf("No elements in stack");
        else
        {
            e = peek();
            printf("\n Top element : %d", e);
        }
        break;
    case 4:
        display();
        break;
    case 5:
        exit(0);
    default :
        printf(" Wrong choice, Please enter correct choice  ");
        break;
    }
  }
}
```

```
/* Count stack elements */
void stack_count()
{
   printf("\n No. of elements in stack : %d", count);
}

/* Push data into stack */
void push(int data)
{
   if (top == NULL)
   {
      top =(struct node *)malloc(1*sizeof(struct node));
      top->ptr = NULL;
      top->info = data;
   }
   else
   {
      temp =(struct node *)malloc(1*sizeof(struct node));
      temp->ptr = top;
      temp->info = data;
      top = temp;
   }
   count++;
}

/* Display stack elements */
void display()
```

```
{
   top1 = top;

   if (top1 == NULL)
   {
      printf("Stack is empty");
      return;
   }

   while (top1 != NULL)
   {
      printf("%d ", top1->info);
      top1 = top1->ptr;
   }
 }

/* Pop Operation on stack */
void pop()
{
   top1 = top;

   if (top1 == NULL)
   {
      printf("\n Error : Trying to pop from empty stack");
      return;
   }
   else
```

```
        top1 = top1->ptr;
    printf("\n Popped value : %d", top->info);
    free(top);
    top = top1;
    count--;
}

/* Return top element */
int peek()
{
    return(top->info);
}
```

7. Write a C program to implement queue using array.

ANSWER:

```
/*
 * C Program to Implement a Queue using an Array
 */
#include <stdio.h>

#define MAX 50

void enqueue();
void dequeue();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
main()
{
    int choice;
    while (1)
    {
        printf("1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("4.Quit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                   enqueue();
                   break;
            case 2:
                  dequeue();
                  break;
            case 3:
                  display();
                  break;
            case 4:
                  exit(1);
            default:
                printf("Wrong choice \n");
        } /* End of switch */
    } /* End of while */
} /* End of main() */

void enqueue()
{
    int add_item;
    if (rear == MAX - 1)
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
        /*If queue is initially empty */
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
    }
} /* End of insert() */

void dequeue()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
} /* End of delete() */

void display()
{
    int i;
    if (front == - 1)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
        printf("\n");
    }
} /* End of display() */
```

8. **Write a C program to test if a string is a palindrome or not using a stack data structure (Note: palindromes ignore spacing, punctuation, and capitalization)**

ANSWER: MADAM

```
// C implementation of the approach
#include <malloc.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 20

int top = -1;

// push function
void push(char ele)
{
        stack[++top] = ele;
}

// pop function
char pop()
{
        return stack[top--];
}

// Function that returns 1
// if str is a palindrome
```

```
int isPalindrome(char str[])
{
    int length = strlen(str);
    int stack[MAX]
    // Finding the mid
    int i, mid = length / 2;

    for (i = 0; i < mid; i++) {
        push(str[i]);
    }

    // Checking if the length of the string is odd, if odd then neglect the middle character
    if (length % 2 != 0) {
        i++;
    }
    // While not the end of the string
    while (str[i] != '\0') {
        char ele = pop();

        // If the characters differ then the
        // given string is not a palindrome
        if (ele != str[i])
            return 0;
        i++;
    }
    return 1;
}
```

```
// Driver code
int main()
{
    char str[] = "madam";

    if (isPalindrome(str)) {
        printf("Yes");
    }
    else {
        printf("No");
    }
    return 0;
}
```

9. **Write a C program that compresses a string by deleting all space characters in the string using queue data structure**

ANSWER:

```
#include <stdio.h>
#define MAX 50
char q[MAX],f=-1,r=-1;

void enq(char val)
{
        if(r == MAX-1)
                printf("queue is full and hence cannot insert");
        else if(f == -1 && r == -1)
                f=r=0;
        else
                r=r+1;
        q[r]=val;
}
char deq()
{
        char val;
        if(f == -1)
                printf("queue is empty and hence cannot delete");
        else
        {
                val = q[f];
                if(f == r)
                        f=r=-1;
                else
                        f=f+1;
        }
        return val;
}
int main()
{
        int i;
        char s[MAX];
        gets(s);
        for(i=0;s[i]!='\0';i++)
        {
                if(s[i] != ' ')
```

```
            enq(s[i]);
    }
    for(i= f;i<=r;i++)
        s[i]=deq();
    s[i]='\0';
    puts(s);

}
```

10. Write an algorithm to convert infix expression to postfix expression. Show stepwise execution of algorithm for converting infix expression to postfix expression for following expression A * B + C * D

11. Write a program to implement Queue using Linked List.

Answer:

```
// A C program to demonstrate linked list based implementation of queue
#include <stdio.h>
#include <stdlib.h>

// A linked list (LL) node to store a queue entry
struct QNode {
        int key;
        struct QNode* next;
};



// The queue, front stores the front node of LL and rear stores the
// last node of LL
struct Queue {
        struct QNode *front, *rear;
};

// A utility function to create a new linked list node.
struct QNode* newNode(int k)
{
        struct QNode* temp = (struct QNode*)malloc(sizeof(struct QNode));
        temp->key = k;
        temp->next = NULL;
        return temp;
}

// A utility function to create an empty queue
struct Queue* createQueue()
{
    struct Queue* q = (struct Queue*)malloc(sizeof(struct Queue));
    q->front = q->rear = NULL;
    return q;
}

// The function to add a key k to q
void enQueue(struct Queue* q, int k)
{
    // Create a new LL node
    struct QNode* temp = newNode(k);

    // If queue is empty, then new node is front and rear both
    if (q->rear == NULL) {
        q->front = q->rear = temp;
        return;
    }

    // Add the new node at the end of queue and change rear
    q->rear->next = temp;
    q->rear = temp;
}
```

```
// Function to remove a key from given queue q
void deQueue(struct Queue* q)
{
    // If queue is empty, return NULL.
    if (q->front == NULL)
        return;

    // Store previous front and move front one node ahead
    struct QNode* temp = q->front;

    q->front = q->front->next;

    // If front becomes NULL, then change rear also as NULL
    if (q->front == NULL)
        q->rear = NULL;

    free(temp);
}

// Driver Program to test above functions
int main()
{
    struct Queue* q = createQueue();
    enQueue(q, 10);
    enQueue(q, 20);
    deQueue(q);
    deQueue(q);
```

```
enQueue(q, 30);
enQueue(q, 40);
enQueue(q, 50);
deQueue(q);
printf("Queue Front : %d \n", q->front->key);
printf("Queue Rear : %d", q->rear->key);
return 0;
```

12. Write a C program to check for balanced for parathesis using stack. Simulate with an example.

ANSWER

```
#include <stdio.h>

#include <stdlib.h>


#define MAX 100


struct stack {

  char stck[MAX];

  int top;

}s;


void push(char item) {

  if (s.top == (MAX - 1))

    printf("Stack is Full\n");


  else {

    s.top = s.top + 1;

    s.stck[s.top] = item;

  }

}


void pop() {

  if (s.top == -1)

    printf("Stack is Empty\n");


  else
```

```
    s.top = s.top - 1;
}

int checkPair(char val1,char val2){
    return (( val1=='(' && val2==')' )||( val1=='[' && val2==']' )||( val1=='{' && val2=='}' ));
}

int checkBalanced(char expr[], int len){

    for (int i = 0; i < len; i++)
    {
        if (expr[i] == '(' || expr[i] == '[' || expr[i] == '{')
        {
          push(expr[i]);
        }
        else
        {
            // exp = {{}}}
            // if you look closely above {{}} will be matched with pair, Thus, stack "Empty"
            //but an extra closing parenthesis like '}' will never be matched
            //so there is no point looking forward
        if (s.top == -1)
            return 0;
        else if(checkPair(s.stck[s.top],expr[i]))
        {
            pop();
            continue;
        }
        // will only come here if stack is not empty
        // pair wasn't found and it's some closing parenthesis
        //Example : {{}}(]
        return 0;
        }
    }
    return 1;
}
int main() {
  char exp[MAX] = "({})[]{}";
  int i = 0;
  s.top = -1;

  int len = strlen(exp);
  checkBalanced(exp, len)?printf("Balanced"): printf("Not Balanced");

  return 0;
}
```

13. Write a C program for Singly Linked list for performing following operations

i. Create SLL
ii. Display SLL
iii. Delete a node from SLL
iv. Append two SLLs

ANSWER:

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

// A linked list (LL) node to store a data element
struct node
{
 int data;
 struct node* next;
}*start;

//Insert data at the beginning
 void insert_at_beg(int x)
 {
  struct node* newnode=(struct node*)malloc(sizeof(struct node));
  printf("\nEnter the data :");
  scanf("%d",&x);
  if(start==NULL)//if LL is empty
  {
    newnode->data=x;
    newnode->next=NULL;
    start=newnode;
  }
  else
  {
   newnode->data=x;
   newnode->next=start;
   start=newnode;
  }
 }

//Delete data from beginning
 void delete_from_beg()
 {
  struct node* ptr;
  if(start==NULL)
  {
   printf("\nUNDERFLOW ! ! !");
   //break;
  }
  else
  {
   ptr=start;
   start=start->next;
   printf("\n%d is deleted...",ptr->data);
   free(ptr);
  }
 }
```

```
void display()
 {
  struct node *ptr=start;
  if(ptr==NULL)
  {
   printf("\nSORRY ! NO ELEMENT ! ! !");
  }
  else
  {
   while(ptr!=NULL)
    {
    printf("%d -> ",ptr->data);
    ptr=ptr->next;
    }
  }
 }

void append()
{
 struct node *head=NULL;
 int m,j,x;
 printf("Create a list:-\n");
 printf("\nEnter the no. of nodes : ");
 scanf("%d",&m);

 for(j=0;j<m;j++)
 {
```

```
 struct node* temp=(struct node*)malloc(sizeof(struct node));
 printf("\nEnter the data %d : ",j+1);
 scanf("%d",&x);
 temp->data=x;
 temp->next=NULL;
 if(head==NULL)
 {
  head=temp;
 }
 else
 {
  struct node *add=head;
  while(add->next!=NULL)
    add=add->next;
  add->next=temp;
 }
}
struct node *ptr1=start,*ptr2=head;
while(ptr1->next!=NULL)
  ptr1=ptr1->next;

ptr1->next=ptr2;//set last node ptr of LL1 to first node of LL2


printf("\n->Concatenated List :-\n");
struct node *k=start;
while(k!=NULL)
   {
    printf("%d ",k->data);
    k=k->next;
   }
}


void main()
{
 start=NULL;
 //head;
 int ch,x,pos;
 int z=1;

while(z)  //While loop to keep program in loop
{
 printf("\n\n----------------------------------------\n");

 printf("1. Insert at the begining\n");
 printf("2. Delete from begining\n");
 printf("3. Append two LL\n");
 printf("4. Display\n");
 printf("5. Exit");

 printf("\n------------------------------------------\n");

 printf("ENTER A CHOICE :");
 scanf("%d",&ch);
 switch(ch)
 {
  case 1: insert_at_beg(x);
          break;
  case 2: delete_from_beg();
          break;
  case 3: append();
          break;
  case 4: display();
          break;
  case 5: exit(0);
          break;
  default : printf("\nOOPS ! WRONG CHOICE !");
            break;
 }
 //goto head;
printf(" \n Do you want to cotin....? press 1 or 0 ");
scanf("%d",&x);
}//end of while loop
}//end of main()
```

14. . Write a C program for Singly Linked list for performing following operations

i. Insert at end
ii. Display
iii. Count odd and even number of elements

ANSWER:

```
#include<stdio.h>

#include<stdlib.h>


// A linked list (LL) node to store a data element

struct node

{

 int data;

 struct node* next;

}*start;


//Insert data at the end

 void insert_at_end(int x)

  {

  struct node* ptr;

  struct node* newnode=(struct node*)malloc(sizeof(struct node));

  printf("\nEnter the data :");

  scanf("%d",&x);

  newnode->data=x;

  newnode->next=NULL;

  if(start==NULL)//if LL is empty

  {  newnode->data=x;

     newnode->next=NULL;
     start=newnode;
  }
  else //if not empty then
  {
   ptr=start;
   while(ptr->next!=NULL)//traverse through LL till last node
     ptr=ptr->next;

   ptr->next=newnode;//link last node of next to newnode
  }
 }


void display()
 {
  struct node *ptr=start;
  if(ptr==NULL)
  {
   printf("\nSORRY ! NO ELEMENT ! ! !");
  }
  else
  {
   while(ptr!=NULL)
   {
    printf("%d -> ",ptr->data);
    ptr=ptr->next;
   }
 }
}

void count_even_odd()
 {
  struct node *ptr=start;
  if(ptr==NULL)
  {
   printf("\nSORRY ! NO ELEMENT ! ! !");
  }
  else
  {
         int even,odd;
   while(ptr!=NULL)
   {
          no=ptr->data;
    if(no%2==0)
          {
                        even=even+1;
    }
    else
          {
                        odd=odd+1;
    }
    ptr=ptr->next;
   }
   printf("No of Even elements %d",even);
   printf("No of Odd elements %d",odd);
  }

 }



void main()
{
 start=NULL;
 //head;
 int ch,x,pos;
 int z=1;

while(z)  //While loop to keep program in loop
{
 printf("\n\n----------------------------------------\n");

 printf("1. Insert at the begining\n");
 printf("2. Display\n");
 printf("3. Count odd and even number of elements\n");
 printf("4. Exit");

 printf("\n------------------------------------------\n");

 printf("ENTER A CHOICE :");
 scanf("%d",&ch);
 switch(ch)
 {
  case 1: insert_at_beg(x);
          break;
  case 2: display();
          break;
  case 3: count_even_odd();
          break;
  case 4: exit(0);
          break;
  default : printf("\nOOPS ! WRONG CHOICE !");
            break;
 }
 //goto head;
printf(" \n Do you want to cotin....? press 1 or 0 ");
scanf("%d",&x);
}//end of while loop
}//end of main()
```

15. Write a function to find and display the sum and average of elements in a singly linked list.

ANSWER:

```
#include<stdio.h>

#include<stdlib.h>


// A linked list (LL) node to store a data element

struct node

{

 int data;

 struct node* next;

}*start;


void sum_avg()

{

  struct node *ptr=start;

  if(ptr==NULL)

  {

   printf("\nSORRY ! NO ELEMENT ! ! !");

  }

  else

  {

           int sum,avg,total_elements;

    total_elements=0;

   while(ptr!=NULL)

   {

            total_elements=total_elements+1;
    sum=sum+(ptr->data);
           }
   avg=sum/total_elements;


   print("Sum %d",sum);
   print("Average %d",avg);
  }
}
void main()
{
 start=NULL;
 //head;
 void sum_avg();


}//end of main()
```

16. Explain Double Ended Queue. Write a c program to implement Double Ended Queue.

```
#include<stdio.h>
#include<stdlib.h>
# define size 40

int front=-1,rear=-1;
int deque[size];
void insert_rear(int max)
{
 int x;
 if((front==0 && rear==max-1)||front==rear+1)
 {
  printf("\nOVERFLOW ! ! !");
 }
 else
 {
  printf("\nEnter the value : ");
  scanf("%d",&x);
  if(front==-1)
    front=rear=0;
  else
  if(rear==max-1)
    rear=0;
  else
   rear++;
  deque[rear]=x;
 }
}

void insert_front(int max)
{
 int x;
 if((front==0 && rear==max-1)||front==rear+1)
 {
  printf("\nOVERFLOW ! ! !");
 }
 else
 {
  printf("\nEnter the value : ");
  scanf("%d",&x);
  if(front==-1)
    front=rear=0;
  else
  if(front==0)
    front=max-1;
  else
   front--;
  deque[front]=x;
 }
}
void delete_front(int max)
{
 if(front==-1)
 {
  printf("\nUNDERFLOW ! ! !");
 }
 else
 {
  printf("\n%d is deleted....",deque[front]);
  if(front==rear)
    front=rear=-1;
  else
  if(front==max-1)
    front=0;
  else
    front++;
 }
}
void delete_rear(int max)
{
 if(front==-1)
 {
  printf("\nUNDERFLOW ! ! !");
 }
 else
 {
  printf("\n%d is deleted....",deque[rear]);
  if(front==rear)
    front=rear=-1;
  else
  if(rear==0)
    rear=max-1;
  else
    rear--;
 }
}
void display(int max)
{
 int f=front,r=rear;
 if(f==-1)
 {
  printf("\nSORRY ! NO ELEMENT ! ! !");
 }
 else
 {
  printf("\n");
  if(f<=r)
  {
   while(f<=r)
   {
    printf("%d ",deque[f]);
    f++;
   }
  }
  else
  {
   while(f<=max-1)
   {
    printf("%d ",deque[f]);
    f++;
   }
   f=0;
   while(f<=rear)
   {
    printf("%d ",deque[f]);
    f++;
   }
  }
 }
}
void input_deque(int max)
{
 int ch;
 do
 {
  printf("\n-----INPUT RESTRICTED DEQUEUE-----");
  printf("\n1. Insert at rear\n2. Delete from front\n3. Delete from rear\n4. Display\n5. Exit");
  printf("\n----------------------------------");
  printf("\nEnter your choice : ");
  scanf("%d",&ch);
  switch(ch)
  {
   case 1: insert_rear(max);
             break;
   case 2: delete_front(max);
            break;
   case 3: delete_rear(max);
            break;
   case 4: display(max);
            break;
   case 5: exit(0);
            break;
   default : printf("\nOOPS ! WRONG INPUT !");
  }
 }while(ch!=5);
}
void output_deque(int max)
{
 int ch;
 do
 {
  printf("\n-----OUTPUT RESTRICTED DEQUEUE-----");
  printf("\n1. Insert at rear\n2. Insert at front\n3. Delete from front\n4. Display\n5. Exit");
  printf("\n---------------------------------");
  printf("\nEnter your choice : ");
  scanf("%d",&ch);
  switch(ch)
  {
   case 1: insert_rear(max);
            break;
   case 2: insert_front(max);
```

```
           break;
   case 3: delete_front(max);
           break;
   case 4: display(max);
           break;
   case 5: exit(0);
           break;
   default : printf("\nOOPS ! WRONG INPUT !");
  }
 }while(ch!=5);
}

void main()
{
 int ch,max;
 printf("Enter the queue capacity : ");
 scanf("%d",&max);
 printf("\n----------MAIN MENU----------\n");
 printf("1. INPUT RESTRICTED DEQUEUE\n2. OUTPUT RESTRICTED DEQUEUE\n");
 printf("\nEnter your choice : ");
 scanf("%d",&ch);
 switch(ch)
 {
  case 1: input_deque(max);
          break;
  case 2: output_deque(max);
          break;
  default : printf("\nOPPS ! WRONG INPUT ! ! !");
 }
}
```

17. Explain with suitable example polynomial representation and addition using linked list.

18. Write a C program for Circular Linked list for performing following operations

i. Insert the node at the beginning
ii. Insert the node at end
iii. Count number of nodes
iv. Display the list

ANSWER:

```
// C program for the above operation

#include <stdio.h>

#include <stdlib.h>


// Structure of a linked list node

struct node {

        int info;

        struct node* next;

};


// Pointer to last node in the list

struct node* last = NULL;


// Function to add a new node at the

// end of the list

void addatlast(int data)

{

        // Initialize a new node

        struct node* temp;

        temp = (struct node*)malloc(sizeof(struct node));
        // If the new node is the
        // only node in the list
        if (last == NULL) {
                temp->info = data;
                temp->next = temp;
                last = temp;
        }

        // Else the new node will be the
        // last node and will contain
        // the reference of head node
        else {
                temp->info = data;
                temp->next = last->next;
                last->next = temp;
                last = temp;
        }
}


// Function to insert a node in the
// starting of the list
void insertAtFront(int data)
{
    // Initialize a new node
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
    // If the new node is the only
    // node in the list
    if (last == NULL) {
        temp->info = data;
        temp->next = temp;
        last = temp;
    }

    // Else last node contains the
    // reference of the new node and
    // new node contains the reference
    // of the previous first node
    else {
        temp->info = data;
        temp->next = last->next;

        // last node now has reference
        // of the new node temp
        last->next = temp;
    }
}

// Function to print the list
void display()
{
          // If list is empty
    if (last == NULL)
        printf("\nList is empty\n");

    // Else print the list
    else {
        struct node* temp;
        temp = last->next;
        do {
            printf("\nData = %d", temp->info);
            temp = temp->next;
        } while (temp != last->next);
    }
}

// Function to print the list
void no_of_nodes()
{
    int nodes=0;
  // If list is empty
    if (last == NULL)
        printf("\nList is empty\n");

    // Else print the list
    else {
        struct node* temp;
        temp = last->next;
        do {
            nodes=nodes+1;
            temp = temp->next;
        } while (temp != last->next);
  printf("\nNo of nodes = %d", nodes);
    }
}

// Driver Code
int main()
{
    // Function Call
    addatlast(10);
    insertAtFront(20);
 void no_of_nodes();
    display();

    return 0;
}
```