VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
On

DATA STRUCTURES (23CS3PCDST)

Submitted by

SONAL(1BM22CS286)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Dec 2023- March 2024

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

This is to certify that the Lab work entitled **"DATA STRUCTURES"** carried out by SONAL **(1BM22CS286)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2023-24. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab - **(23CS3PCDST)**work prescribed for the said degree.

Prof. Lakshmi Neelima
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

Index Sheet

Sl. No.	Experiment Title	Page No.
1	**Lab program 1**	4
2	**Lab program 2**	7
3	**Lab program 3**	10
4	**Lab program 4**	13
5	**Lab program 5**	17
6	**Lab program 6**	25
7	**Lab program 7**	27
8	**Lab program 8**	31
9	**Leetcode Program**	34
10	**Lab program 9**	37
11	**Lab program 10**	39
12	**Lab program 11**	40
13	**Leetcode Program**	42

Course outcomes:

CO1	Apply the concept of linear and nonlinear data structures.
CO2	Analyze data structure operations for a given problem
CO3	Design and develop solutions using the operations of linear and nonlinear data structure for a given specification.
CO4	Conduct practical experiments for demonstrating the operations of different data structures.

Lab program 1:

Write a program to simulate the working of stack using an array with the following:
a) Push
b) Pop
c) Display
The program should print appropriate messages for stack overflow, stack underflow.

```
#include<stdio.h>
#include<stdlib.h>
int stack[5];
int top=-1;

void push()
{
   if(top==4)
   {
      printf("stack overflow\n");
   }
  else{
   int n;
   printf("enter the value to be pushed\n");
   scanf("%d",&n);
   top++;
   stack[top]=n;
   printf("insertion sucessfull");

 }
}

 void pop()
 {
    if(top==-1)
    {
       printf("stack underflow\n");
```

```
    }
    else{
       int n;
       n=stack[top];
       printf("deleted element is %d \n",n);
       top--;
    }
}

void display()
{
    if(top==-1)
    {
        printf("stack is empty\n");
    }

    for(int i=top;i>=0;i--)
    {
        printf("%d\n",stack[i]);
    }
}


void main()
{
    int choice;
     while(1)
     {
    printf("enter the choice\n");
    scanf("%d",&choice);
     printf("Enter 1.push 2.pop 3.display 4.exit\n");
        switch(choice)
        {
            case 1:push();
            break;
            case 2:pop();
            break;
            case 3:display();
            break;
            case 4:exit(0);
            break;

            default:
               printf("invalid option");
        }
    }
}
```

Output:

```
enter the choice
1
Enter 1.push 2.pop 3.display 4.exit
enter the value to be pushed
1
insertion sucessfullenter the choice
1
Enter 1.push 2.pop 3.display 4.exit
enter the value to be pushed
2
insertion sucessfullenter the choice
1
Enter 1.push 2.pop 3.display 4.exit
enter the value to be pushed
3
insertion sucessfullenter the choice
1
Enter 1.push 2.pop 3.display 4.exit
enter the value to be pushed
4
insertion sucessfullenter the choice
1
Enter 1.push 2.pop 3.display 4.exit
enter the value to be pushed
5
insertion sucessfullenter the choice
3
Enter 1.push 2.pop 3.display 4.exit
5
4
```

```
3
2
1
enter the choice
1
Enter 1.push 2.pop 3.display 4.exit
stack overflow
enter the choice
2
Enter 1.push 2.pop 3.display 4.exit
deleted element is 5
enter the choice
2
Enter 1.push 2.pop 3.display 4.exit
deleted element is 4
enter the choice
2
Enter 1.push 2.pop 3.display 4.exit
deleted element is 3
enter the choice
2
Enter 1.push 2.pop 3.display 4.exit
deleted element is 2
enter the choice
2
Enter 1.push 2.pop 3.display 4.exit
deleted element is 1
enter the choice
2
Enter 1.push 2.pop 3.display 4.exit
Enter 1.push 2.pop 3.display 4.exit
stack underflow
enter the choice
```

Lab program 2:

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus) (multiply) and /(divide)

```
#include <stdio.h>
#include <string.h>
int index1=0,pos=0,top=-1,length;
char symbol,temp,infix[50],postfix[50],stack[50];
void infixtopostfix();
void push(char symbol);
char pop();
int preced(char symbol);

void main()
{
```

```
   printf("Enter the infix expression\n");
   scanf("%s",infix);
   infixtopostfix();
   printf("infix expression is %s\n",infix);
   printf("postfix expression is %s\n",postfix);

}

void infixtopostfix()
{
   length=strlen(infix);
   push('#');
   while(index1<length)
   {
      symbol=infix[index1];
      switch(symbol)
      {
         case '(' : push(symbol);
            break;
            case ')' : temp=pop();
            while(temp!='(')
            {
               postfix[pos]=temp;
               pos++;
               temp=pop();
            }
            break;
            case '+' :
            case '-' :
            case '*':
            case '/':
               case '^' : while(preced(stack[top])>=preced(symbol))
               {
                  temp=pop();
                  postfix[pos]=temp;
                  pos++;
               }
                  push(symbol);
                  break;

      default : postfix[pos++]=symbol;
      }
      index1++;
   }
   while(top>0)
   {
      temp=pop();
      postfix[pos]=temp;
      pos++;
   }
```

```
}

void push(char symbol)
{
   top++;
   stack[top]=symbol;
}

char pop()
{
   char symb;
   symb=stack[top];
   top--;
   return (symb);
}

int preced(char symbol)
{
   int p;
   switch(symbol)
   {
      case '^' : p=3;
      break;
      case '*' :
         case '/' : p=2;
         break;
         case '+' :
            case '-' : p=1;
            break;
            case '(' : p=0;
            break;
            case '#' :p=-1;

   }
   return (p);
}
```

```
Enter the infix expression
(A+(B*C-(D/E^F)*G)*H)
infix expression is (A+(B*C-(D/E^F)*G)*H)
postfix expression is ABC*DEF^/G*-H*+

Process returned 38 (0x26)   execution time : 39.093 s
Press any key to continue.
```

Lab program 3:

Write a program to simulate the working of the queue of integers using an array. Provide the following operations: Insert, delete, display. The program should print appropriate message for overflow and underflow condition.

```
#include<stdio.h>
#include<stdlib.h>
int q[5];
int f=-1,r=-1;

void enqueue()
{
   int n;
   printf("enter the element\n");
   scanf("%d",&n);
   if(r==4)
   {
      printf("queue is full\n");
   }
   else if(f==-1&&r==-1)
   {
      f=0;
      r=0;
      q[r]=n;
   }
   else{
      r++;
      q[r]=n;
   }
}

void dequeue()
{
  int n;
   if (f == -1) {
      printf("Queue is empty\n");
   } else {
      n = q[f];
      printf("The dequeued element is %d\n", n);
      if (f == r) {
         f = -1;
         r = -1;
      } else {
         f++;
      }  }}
void display()
{
    if(f==-1)
   {
      printf("queue is empty\n");
```

```
        }
        else{

    for(int i=f;i<=r;i++)
    {
        printf("%d\t",q[i]);
    }
}
}

void main()
{
    int choice;
      while(1)
      {
     printf("enter the choice\n");
     scanf("%d",&choice);
      printf("Enter 1.enqueue 2.dequeue 3.display 4.exit\n");
         switch(choice)
         {
             case 1:enqueue();
             break;
             case 2:dequeue();
             break;
             case 3:display();
             break;
             case 4:exit(0);
             break;
             default:
 printf("invalid option");
         }
 }
 }
```

```
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
1
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
2
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
3
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
4
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
5
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
6
```

```
queue is full
enter the choice
3
Enter 1.enqueue 2.dequeue 3.display 4.exit
1       2       3       4       5       enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
The dequeued element is 1
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
The dequeued element is 2
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
The dequeued element is 3
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
The dequeued element is 4
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
The dequeued element is 5
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
Queue is empty
enter the choice
```

Lab program 4:

Write a program to simulate the working of a circular queue using an array. Provide the following operations: insert, delete& display. The program should print appropriate message for queue empty and queue overflow conditions.

```
#include<stdio.h>
#include<stdlib.h>
int q[5];
int f=-1,r=-1;
void enqueue()
{
   int n;
   printf("enter the element\n");
   scanf("%d",&n);
   if(f==-1&&r==-1)
   {
      f=0;
      r=0;
      q[r]=n;
   }
   else if(((r+1)%5)==f)
   {
      printf("queue is full");
   }
   else{
      r=(r+1)%5;
      q[r]=n;
   }
}


void dequeue()
{
   if(f==-1&&r==-1)
   {
      printf("queue is empty");
 }
   else if(f==r)
   {
      printf("the deleted element is %d\n",q[f]);
      f=-1;
      r=-1;
   }
   else{
       printf("the deleted element is %d\n",q[f]);
       f=(f+1)%5;
   }
}
void display()
{
```

```
   int i;
   if(f==-1&&r==-1)
   {
      printf("queue is empty");
 }
   else{
      for( i=f;i!=r;i=(i+1)%5)
      {
         printf("%d",q[i]);
      }
      printf("%d",q[i]);
}
}
void main()
{
   int choice;
     while(1)
     {
    printf("enter the choice\n");
    scanf("%d",&choice);
     printf("Enter 1.enqueue 2.dequeue 3.display 4.exit\n");
        switch(choice)
        {
           case 1:enqueue();
           break;
           case 2:dequeue();
           break;
           case 3:display();
           break;
           case 4:exit(0);
           break;
          default:
              printf("invalid option");
        }
    }
 }
```

```
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
1
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
2
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
3
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
4
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
5
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
6
```

```
queue is fullenter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
the deleted element is 1
enter the choice
1
Enter 1.enqueue 2.dequeue 3.display 4.exit
enter the element
10
enter the choice
3
Enter 1.enqueue 2.dequeue 3.display 4.exit
234510enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
the deleted element is 2
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
the deleted element is 3
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
the deleted element is 4
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
the deleted element is 5
enter the choice
2
```

```
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
the deleted element is 10
enter the choice
2
Enter 1.enqueue 2.dequeue 3.display 4.exit
queue is emptyenter the choice
```

Lab program 5:

WAP to Implement Singly Linked List with following operations.
a) Create a linked list.
b) Insertion of a node at first position, at any position and at end of list.
c) Display the contents of the linked list.
d) Deletion of first element, specified element and last element in the list.
e) sort,reverse,concatination

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

struct node
{
   int data;
   struct node *next;
}*head,*newnode,*temp,*head2;

void createll1()
{
  int n;
  printf("Enter the data\n");
  scanf("%d",&n);
  newnode=(struct node*)malloc(sizeof(struct node));
  newnode->data=n;
  if(head==NULL)
  {
     head=temp=newnode;
  }
  else{
   temp->next=newnode;
   temp=newnode;
   newnode->next=NULL
 }

}

void createll2()
{
  int n;
  printf("Enter the data\n");
  scanf("%d",&n);
  newnode=(struct node*)malloc(sizeof(struct node));
  newnode->data=n;
  if(head2==NULL)
  {
     head2=temp=newnode;
  }
  else{
   temp->next=newnode;
```

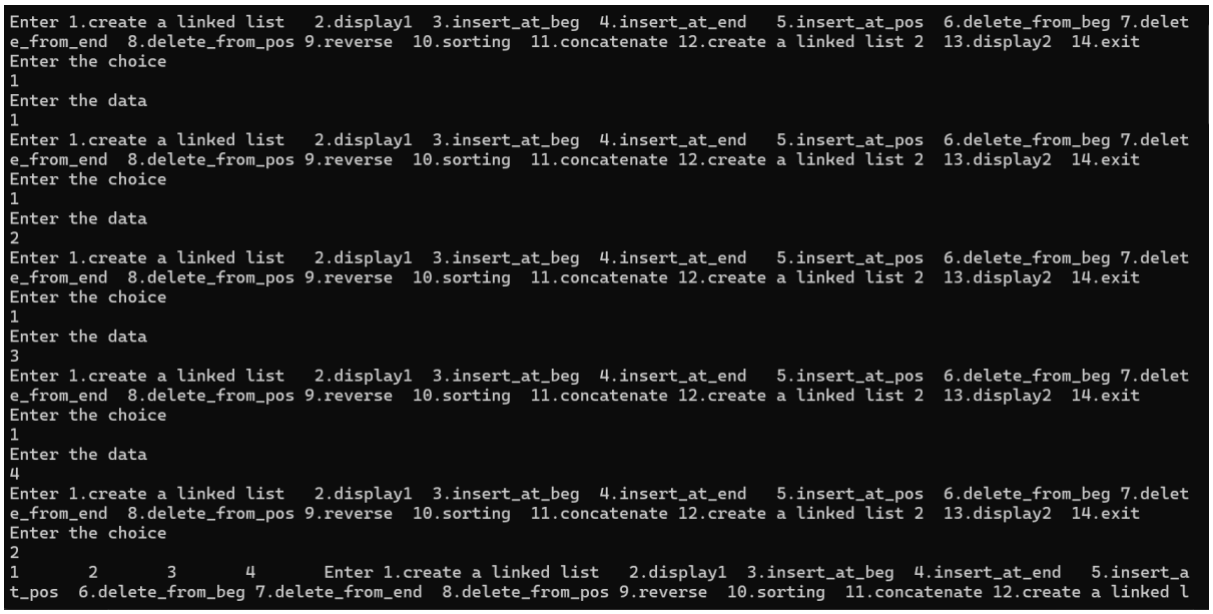
```
   temp=newnode;
   newnode->next=NULL;
  }

}

void display1()
{

   temp=head;
   while(temp!=0)
   {
      printf("%d \t",temp->data);
      temp=temp->next;
   }

}

void display2()
{
    temp=head2;
   while(temp!=0)
   {
      printf("%d \t",temp->data);
      temp=temp->next;
   }
}




void insert_at_beg()
{
   int n;
   printf("Enter the data to be entered\n");
   scanf("%d",&n);
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=n;
   newnode->next=head;
   head=newnode;
   printf("Insertion successful\n");

}

void insert_at_end()
{
   int n;
   printf("Enter the data to be entered\n");
   scanf("%d",&n);
   newnode=(struct node*)malloc(sizeof(struct node));
```

```
   newnode->data=n;
   temp=head;
   while(temp->next!=NULL)
   {
      temp=temp->next;
   }
   temp->next=newnode;
   newnode->next=NULL;

}

void insert_at_pos()
{
   int n,i=1,pos;
   printf("Enter the data to be entered\n");
   scanf("%d",&n);
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=n;
   printf("Enter the pos after which the newnode is to be inserted\n");
   scanf("%d",&pos);
   temp=head;
   while(i<pos)
   {
      temp=temp->next;
      i++;
   }
   newnode->next=temp->next;
   temp->next=newnode;


}

void delete_from_beg()
{
   temp=head;
   head=head->next;
   free(temp);
}

void delete_from_end()
{
   struct node *prevnode;
   temp=head;
   while(temp->next!=NULL)
   {
      prevnode=temp;
      temp=temp->next;
   }
   if(temp==head)
   {
```

```
        head=0;
    }
    else{
        prevnode->next=NULL;
    }
    free(temp);
}

void delete_from_pos()
{
    struct node *nextnode,*node;
    int pos,i=1;
    printf("Enter the pos\n");
    scanf("%d",&pos);
    temp=head;
    while(i<pos)
    {
        node=temp;
        temp=temp->next;
        nextnode=temp->next;
        i++;
    }
    node->next=nextnode;
    free(temp);
}


void reverse()
{
    struct node *prevnode,*nextnode;
    temp=nextnode=head;
    prevnode=NULL;
    while(temp!=NULL)
    {
        temp=temp->next;
        nextnode->next=prevnode;
        prevnode=nextnode;
        nextnode=temp;
    }
    head=prevnode;
}



void sorting()
{
    int c;
    struct node *nextnode;
    temp=nextnode=head;
    while(temp->next!=NULL)
```

```
   {
      nextnode=temp->next;
      while(nextnode!=NULL)
      {
         if(temp->data>nextnode->data)
         {
            c=temp->data;
            temp->data=nextnode->data;
            nextnode->data=c;
         }
         nextnode=nextnode->next;
      }
      temp=temp->next;
   }
}

void concatenate()
{
   temp=head;
   while(temp->next!=NULL)
   {
      temp=temp->next;
   }


   temp->next=head2;

}

void main()
{

   int choice;
   while(1)
   {
   printf("Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end
5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse
10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit\n");
   printf("Enter the choice\n");
   scanf("%d",&choice);
   switch(choice)
   {
      case 1:createll1();
      break;
      case 2:display1();
      break;
      case 3:insert_at_beg();
      break;
      case 4:insert_at_end();
        break;
```

```
        case 5:insert_at_pos();
        break;
        case 6:delete_from_beg();
        break;
        case 7:delete_from_end();
        break;
        case 8:delete_from_pos();
        break;
        case 9:reverse();
        break;
        case 10:sorting();
        break;
        case 11:concatenate();
        break;
        case 12:createll2();
        break;
        case 13:display2();
        break;
        case 14:exit(0);
        break;

      default :
          printf("invalid input\n");
   }}}
```



```
t_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked l
ist 2  13.display2  14.exit
Enter the choice
3
Enter the data to be entered
5
Insertion successful
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delet
e_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
5       1       2       3       4       Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.
insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a
linked list 2  13.display2  14.exit
Enter the choice
4
Enter the data to be entered
10
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delet
e_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
5       1       2       3       4       10      Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_
end   5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.c
reate a linked list 2  13.display2  14.exit
Enter the choice
5
Enter the data to be entered
7
Enter the pos after which the newnode is to be inserted
```

```
Enter the pos after which the newnode is to be inserted
4
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delet
e_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
5       1       2       3       7       4       10      Enter 1.create a linked list   2.display1  3.insert_at_beg  4.in
sert_at_end   5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concaten
ate 12.create a linked list 2  13.display2  14.exit
Enter the choice
6
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delet
e_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
1       2       3       7       4       10      Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_
end   5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.c
reate a linked list 2  13.display2  14.exit
Enter the choice
7
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delet
e_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
1       2       3       7       4       Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.
insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a
linked list 2  13.display2  14.exit
Enter the choice
8
Enter the pos
```

```
Enter the pos
3
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delet
e_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
1       2       7       4       Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_a
t_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked l
ist 2  13.display2  14.exit
Enter the choice
9
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delet
e_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
4       7       2       1       Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_a
t_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked l
ist 2  13.display2  14.exit
Enter the choice
10
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delet
e_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
1       2       4       7       Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_a
t_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked l
ist 2  13.display2  14.exit
Enter the choice
12
Enter the data
```

```
Enter the data
8
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete
_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
12
Enter the data
9
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete
_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
12
Enter the data
10
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete
_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
12
Enter the data
11
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete
_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
13
8       9       10      11      Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from
_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
11
Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete
_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
13
8       9       10      11      Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end   5.insert_at_pos  6.delete_from
_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  13.display2  14.exit
Enter the choice
2
```

```
1       2       4       7       8       9       10      11      Enter 1.create a linked list   2.display1  3.insert_at_beg  4.insert_at_end
  5.insert_at_pos  6.delete_from_beg 7.delete_from_end  8.delete_from_pos 9.reverse  10.sorting  11.concatenate 12.create a linked list 2  1
3.display2  14.exit
Enter the choice
```

Lab program 6:

Stack implementation using single linked list

```
#include<stdio.h>
#include<math.h>
struct node
{
   int data;
   struct node *next;
}*top;
void push()
{
   int n;
   struct node *newnode;
   printf("Enter the data\n");
   scanf("%d",&n);
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=n;
   newnode->next=top;
   top=newnode;
}
void display()
{
   struct node *temp;
   temp=top;
   while(temp!=0)
   {
      printf("%d\t",temp->data);
      temp=temp->next;
   }}
 void pop()
 {
    struct node *temp,*nextnode;
    temp=top;
    if(temp==NULL)
    {
       printf("underflow");
    }
    else{
   nextnode=temp->next;
    top=nextnode;
    free(temp);
    }
}
void main()
{
   int choice;
   while(1)
   {
   printf("Enter the choice\n");
```

```
   scanf("%d",&choice);
   printf("Enter 1.push 2.display   3.pop   4.exit\n");
   switch(choice)
   {
      case 1:push();
      break;
      case 2:display();
      break;
      case 3:pop();
      break;
      case 4:exit(0);
      break;
      default :
         printf("invalid input\n");
   }
}
}
```

```
Enter the choice
1
Enter 1.push 2.display   3.pop   4.exit
Enter the data
1
Enter the choice
1
Enter 1.push 2.display   3.pop   4.exit
Enter the data
2
Enter the choice
1
Enter 1.push 2.display   3.pop   4.exit
Enter the data
3
Enter the choice
2
Enter 1.push 2.display   3.pop   4.exit
3       2       1       Enter the choice
3
Enter 1.push 2.display   3.pop   4.exit
Enter the choice
3
Enter 1.push 2.display   3.pop   4.exit
Enter the choice
2
Enter 1.push 2.display   3.pop   4.exit
1       Enter the choice
3
Enter 1.push 2.display   3.pop   4.exit
```

```
Enter 1.push 2.display   3.pop   4.exit
Enter the choice
3
Enter 1.push 2.display   3.pop   4.exit
underflowEnter the choice
```

Lab program 7:

Queue implementation using single linked list

```
#include<stdio.h>
#include<math.h>
struct node
{
   int data;
   struct node *next;
}*front,*rear,*temp;
void insert()
{
    int n;
     struct node *newnode;
   printf("Enter the data\n");
   scanf("%d",&n);
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=n;
   newnode->next=NULL;
   if(newnode==NULL)
   {
      printf("overflow");
   }
   else if(front==0&&rear==0)
   {
      front=rear=newnode;
   }
   else{
      rear->next=newnode;
 rear=newnode;
   }
}
void display()
{
 temp=front;
   while(temp!=NULL)
   {
      printf("%d\t",temp->data);
      temp=temp->next;
   }
}
void delete()
{
   if(front==NULL)
   {
      printf("underflow");
   }
   else{
```

```
    temp=front;
    front=temp->next;
    free(temp);
    }
}
void main()
{
     int choice;
     while (1)
    {
        printf("\n1.insert\n");
        printf("2. display\n");
        printf("3. delete\n");
        printf("4. exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
               insert();
               break;
            case 2:
                display();
                break;
            case 3:
                delete();
                break;
             case 4:
                exit(0);
                break;
            default:
                printf("Invalid choice\n");
        }
    }
}
```

```
1.insert
2. display
3. delete
4. exit
Enter your choice: 1
Enter the data
1

1.insert
2. display
3. delete
4. exit
Enter your choice: 1
Enter the data
2

1.insert
2. display
3. delete
4. exit
Enter your choice: 1
Enter the data
3

1.insert
2. display
3. delete
4. exit
Enter your choice: 1
```

```
Enter your choice: 1
Enter the data
4

1.insert
2. display
3. delete
4. exit
Enter your choice: 2
1       2       3       4
1.insert
2. display
3. delete
4. exit
Enter your choice: 3

1.insert
2. display
3. delete
4. exit
Enter your choice: 3

1.insert
2. display
3. delete
4. exit
Enter your choice: 3

1.insert
2. display
```

```
1.insert
2. display
3. delete
4. exit
Enter your choice: 3

1.insert
2. display
3. delete
4. exit
Enter your choice: 3

1.insert
2. display
3. delete
4. exit
Enter your choice: 2

1.insert
2. display
3. delete
4. exit
Enter your choice: 3
underflow
1.insert
2. display
3. delete
4. exit
Enter your choice:
```

Lab program 8:

WAP to Implement doubly link list with primitive operations
a) Create a doubly linked list.
b) Insert a new node to the left of the node.
c) Delete the node based on a specific value

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

struct node
{
   int data;
   struct node *next;
   struct node *prev;
}*head,*temp,*newnode;

void create_dll()
{
   struct node *newnode;
  int n;
  printf("Enter the data\n");
  scanf("%d",&n);
  newnode=(struct node*)malloc(sizeof(struct node));
  newnode->data=n;
  newnode->next=NULL;
  newnode->prev=NULL;
  if(head==0)
  {
     temp=head=newnode;
  }
  else{
   temp->next=newnode;
   newnode->prev=temp;
   temp=newnode;
   newnode->next=NULL;
  }
}


void display()
{
   temp=head;
   while(temp!=0)
   {
      printf("%d\t",temp->data);
      temp=temp->next;
   }
}
```

```
void insert_before()
{
   struct node *prevnode;
   int n,i=1,pos;
   printf("Enter the data to be entered\n");
   scanf("%d",&n);
   newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=n;
   printf("Enter the pos before which the newnode is to be inserted\n");
   scanf("%d",&pos);
   temp=head;
    while(i<pos)
    {
       prevnode=temp;
       temp=temp->next;
       i++;
    }
    newnode->next=temp;
    newnode->prev=prevnode;
    temp->prev=newnode;
    prevnode->next=newnode;

}

void delete_from_specified_position()
{
    struct node *nextnode,*node;
   int pos,i=1;
   printf("Enter the pos\n");
   scanf("%d",&pos);
   temp=head;
   while(i<pos)
   {
      node=temp;
      temp=temp->next;
      nextnode=temp->next;
      i++;
   }
   node->next=nextnode;
   nextnode->prev=node;
   free(temp);

}
```

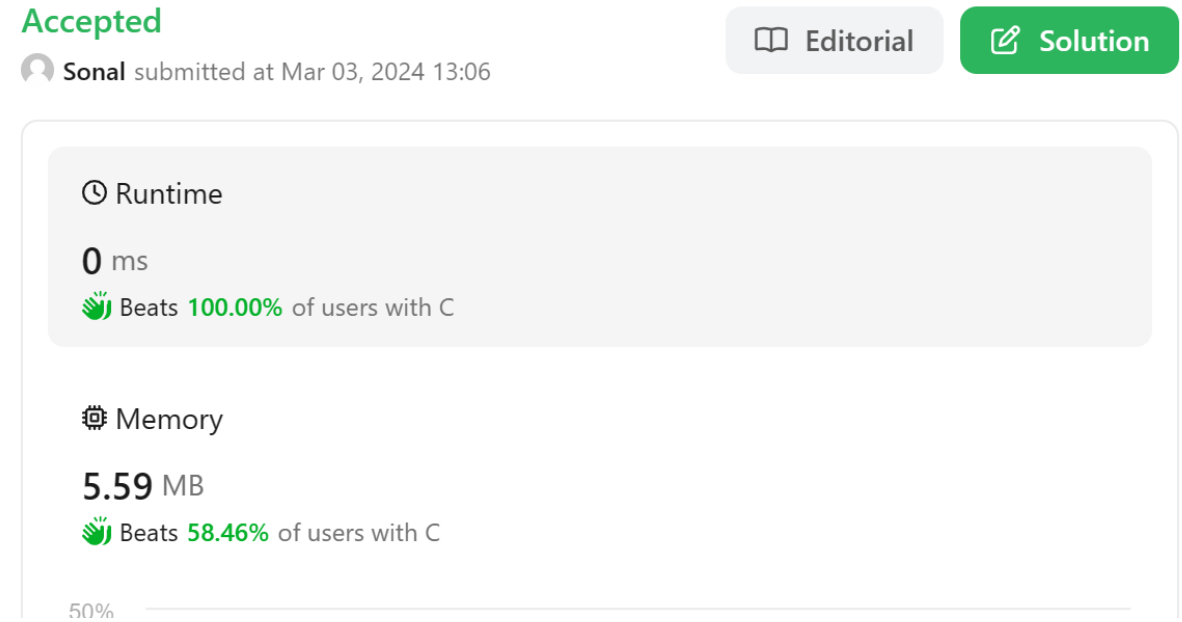
```
void main()
{
    int choice;
   while(1)
   {
   printf("Enter the choice\n");
   scanf("%d",&choice);
   printf("Enter 1.create a  doubly linked list   2.display  3.insert_before
4.delete_from_specified_position   5.exit\n");
   switch(choice)
   {
      case 1:create_dll();
      break;
      case 2:display();
      break;
      case 3:insert_before();
        break;
        case 4:delete_from_specified_position();
        break;
        case 5:exit(0);
        break;


      default :
         printf("invalid input\n");
   }}}
```

```
Enter the choice
1
Enter 1.create a  doubly linked list   2.display  3.insert_before  4.delete_from_specified_position   5.exit
Enter the data
1
Enter the choice
1
Enter 1.create a  doubly linked list   2.display  3.insert_before  4.delete_from_specified_position   5.exit
Enter the data
2
Enter the choice
1
Enter 1.create a  doubly linked list   2.display  3.insert_before  4.delete_from_specified_position   5.exit
Enter the data
3
Enter the choice
2
Enter 1.create a  doubly linked list   2.display  3.insert_before  4.delete_from_specified_position   5.exit
1       2       3       Enter the choice
3
Enter 1.create a  doubly linked list   2.display  3.insert_before  4.delete_from_specified_position   5.exit
Enter the data to be entered
10
Enter the pos before which the newnode is to be inserted
2
Enter the choice
2
Enter 1.create a  doubly linked list   2.display  3.insert_before  4.delete_from_specified_position   5.exit
1       10      2       3       Enter the choice
4
```

```
Enter 1.create a  doubly linked list   2.display  3.insert_before  4.delete_from_specified_position   5.exit
Enter the pos
3
Enter the choice
2
Enter 1.create a  doubly linked list   2.display  3.insert_before  4.delete_from_specified_position   5.exit
1       10      3       Enter the choice
```

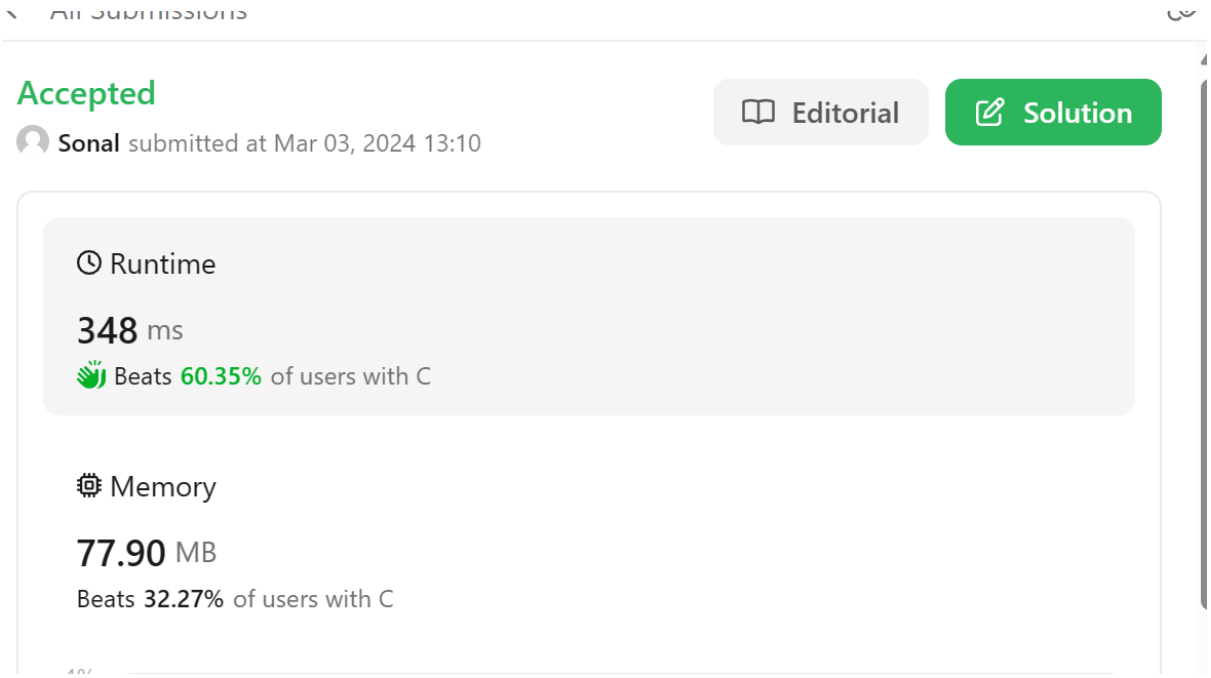
Leetcode Program:
Score of parenthesis

```c
int scoreOfParentheses(char* s) {
    int score = 0;
    int depth = 0;

    for (int i = 0; s[i] != '\0'; i++) {
        if (s[i] == '(') {
            depth++;
        } else {
            depth--;
            if (s[i - 1] == '(') {
                score += 1 << depth;
            }
        }
    }

    return score;
}
```



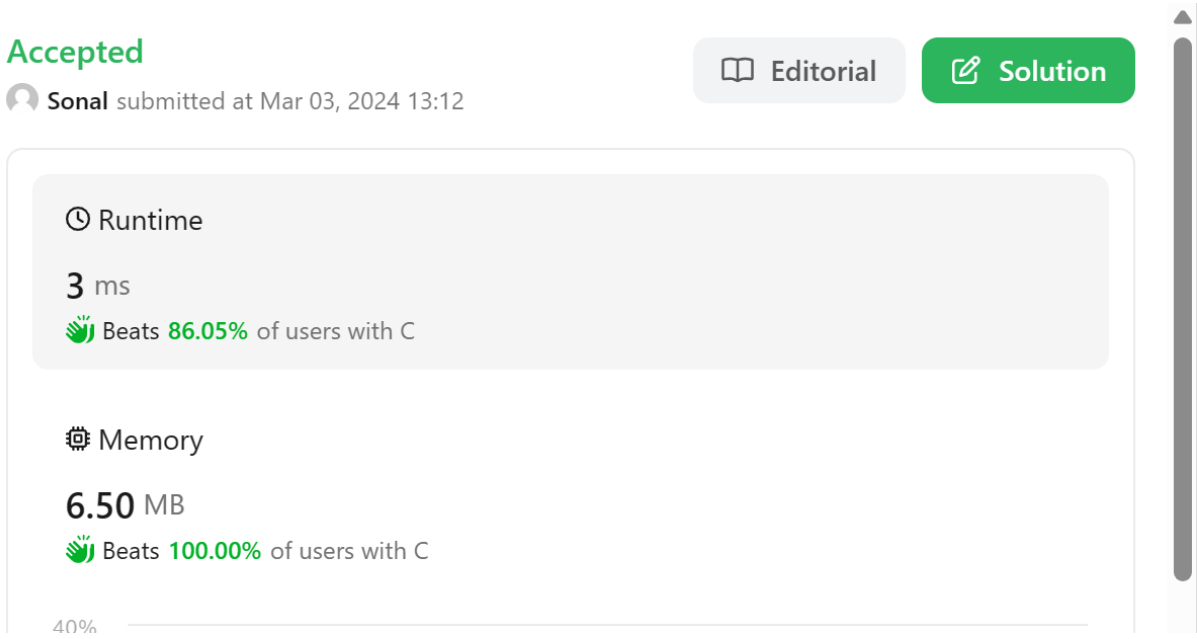
Leetcode Program:
Delete the middle node of linked list

```
struct ListNode* deleteMiddle(struct ListNode* head) {
    int count=0,middlenode,i=0;
    struct ListNode *temp=head;
     struct ListNode *node=head;
      struct ListNode *prevnode;
       struct ListNode *nextnode=head;
    while(temp!=NULL)
    {
        count=count+1;
        temp=temp->next;
    }
    middlenode=count/2;
    if (middlenode == 0) {
        struct ListNode* newHead = head->next;
        free(head);
        return newHead;
    }
    while(i<middlenode)
    {
      prevnode=node;
      node=node->next;
      nextnode=node->next;
      i++;
    }
    prevnode->next=nextnode;
    free(node);
    return head;
}
```



Leetcode Program:
Odd even linked list

```
struct ListNode* oddEvenList(struct ListNode* head) {
    if(head==NULL||head->next==NULL)
    {
        return head;
    }

     struct ListNode* oddtemp=head;
      struct ListNode* eventemp=head->next;
      struct ListNode* evenhead=eventemp;
     while (eventemp != NULL && eventemp->next != NULL)
      {
          oddtemp->next=eventemp->next;
          oddtemp=oddtemp->next;
          eventemp->next=oddtemp->next;
          eventemp=eventemp->next;
      }


        oddtemp->next=evenhead;

      return head;
}
```



Lab Program 9:
Write a program.
a. To construct Binary Search tree
b. Traverse the tree using inorder , postorder, preorder.
c. Display the elements in the tree.

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

struct bstnode
{
   int data;
   struct bstnode *left;
   struct bstnode *right;
}*root,*newnode;

 struct bstnode* getnewnode(int item)
     {
        newnode=(struct bstnode*)malloc(sizeof(struct bstnode));
        newnode->data=item;
        newnode->left=newnode->right=NULL;
        return newnode;
     }

struct bstnode* insert(struct bstnode* root,int data)
{
   if (root==NULL)
   {
      root = getnewnode(data);
   }
   else if(data<=root->data)
   {
      root->left=insert(root->left,data);
   }
   else{
      root->right=insert(root->right,data);
   }
   return root;
}

void inorder(struct bstnode* root)
{
   if(root!=NULL)
   {
```

```
        inorder(root->left);
        printf("%d",root->data);
        inorder(root->right);
    }
}

void preorder(struct bstnode* root)
{
    if(root!=NULL)
    {

        printf("%d",root->data);
         preorder(root->left);
        preorder(root->right);
    }
}
void postorder(struct bstnode* root)
{
    if(root!=NULL)
    {


         postorder(root->left);
        postorder(root->right);
        printf("%d",root->data);
    }
}

void main()
{
    root=NULL;
    root=insert(root, 10);
    root=insert(root, 20);
    root=insert(root, 30);
    root=insert(root, 25);
    root=insert(root, 8);
    root=insert(root, 12);
    root=insert(root, 16);
    printf("insertion successfull\n");
    inorder(root);
    printf("\n");
    preorder(root);
    printf("\n");
    postorder(root);
```

```
    printf("\n");

}
```

```
insertion successfull
8 10 12 16 20 25 30
10 8 20 12 16 30 25
8 16 12 25 30 20 10

Process returned 10 (0xA)   execution time : 1.517 s
Press any key to continue.
```

Lab Program 10:
Breadth First Search

```
#include <stdio.h>
void main()
{
  int am[10][10],n;
  printf("enter no. of nodes:");
  scanf("%d",&n);
  printf("enter the adjacency matrix:");
  for(int i=0;i<n;i++)
      for(int j=0;j<n;j++)
          scanf("%d",&am[i][j]);
  for(int source=0;source<n;source++)
  {
     bfs(am,source,n);
  }
}

void bfs(int am[10][10],int source,int n)
{
   int v[10]={0},q[10],f=0,r=-1;
   q[++r]=source;
   v[source]=1;
   printf("the bfs traversal from %d node is:",source);
   while(f<=r)
```

```
    {
        int u;
        u=q[f++];
        printf("%d",u);
        for(int i=0;i<n;i++)
        {
            if(am[u][i]==1 && v[i]==0)
            {
                q[++r]=i;
                v[i]=1;
            }
        }
    }
    printf("\n");
}
```

```
enter no. of nodes:3
enter the adjacency matrix:0 1 1
1 0 0
1 0 0
the bfs traversal from 0 node is:012
the bfs traversal from 1 node is:102
the bfs traversal from 2 node is:201
```

Lab Program 11:
Depth First Search

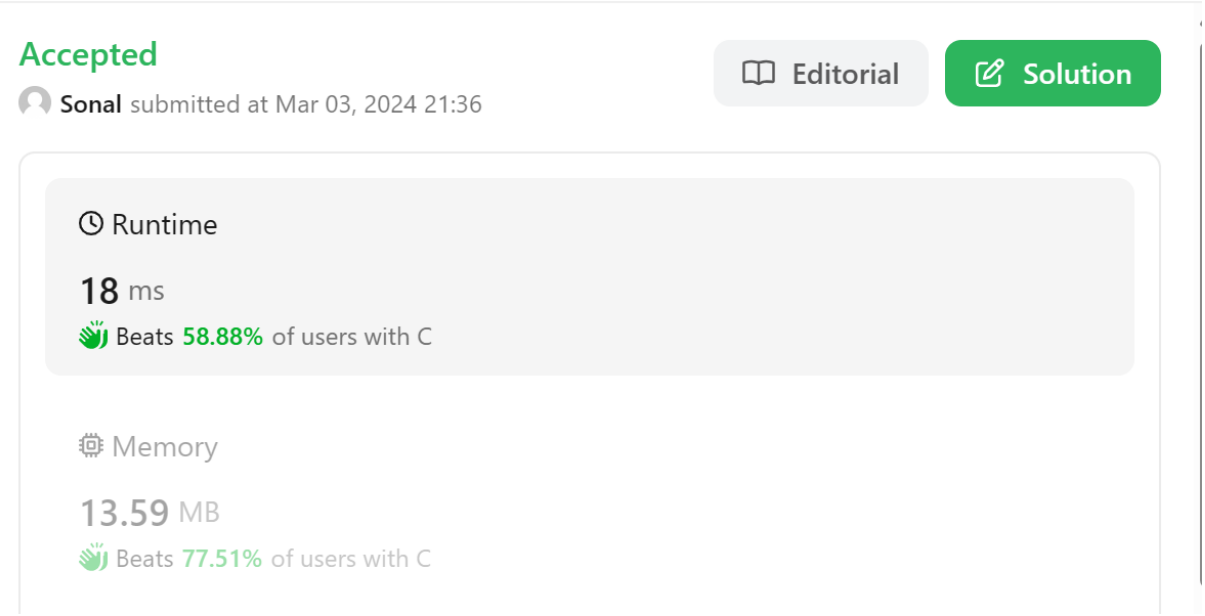
```
#include<stdio.h>
#include<conio.h>
int a[20][20], s[20], n;
void dfs(int v)
{
    int i;
    s[v]=1;
    for(i=1; i<=n; i++)
        if(a[v][i] && !s[i])
        {
            printf("\n %d->%d",v,i);
            dfs(i);
        }
}
```

```
int main()
{
   int i, j, count=0;
   printf("\n Enter number of vertices:");
   scanf("%d", &n);
   for(i=1; i<=n; i++)
   {
         s[i]=0;
         for(j=1; j<=n; j++)
            a[i][j]=0;
   }
   printf("Enter the adjacency matrix:\n");
   for(i=1; i<=n; i++)
         for(j=1; j<=n; j++)
            scanf("%d", &a[i][j]);
   dfs(1);
   printf("\n");
   for(i=1; i<=n; i++)
{
   if(s[i])
         count++;
}

if(count==n)
   printf("Graph is connected");
else
   printf("Graph is not connected");
return 0;
}
```

```
 Enter number of vertices:3
Enter the adjacency matrix:
0 1 1
1 0 0
1 0 0

 1->2
 1->3
Graph is connected
Process returned 0 (0x0)   execution time : 12.206 s
Press any key to continue.
```

Leetcode Program:
Delete node in a BST

```
struct TreeNode* findMin(struct TreeNode* node) {
    while (node->left != NULL) {
        node = node->left;
    }
    return node;
}
struct TreeNode* deleteNode(struct TreeNode* root, int key) {
     if(root==NULL)
    {
        return root;
    }
    else if(key < root->val)
    {
        root->left=deleteNode(root->left,key);
    }
    else if(key > root->val)
    {
        root->right=deleteNode(root->right,key);
    }
    else
    {
        if(root->left==NULL && root->right==NULL)
        {
            free(root);
            root=NULL;
        }
        else if(root->left==NULL)
        {
            struct TreeNode *ptr=root;
            root=root->right;
            free(ptr);
        }
        else if(root->right==NULL)
        {
            struct TreeNode *ptr=root;
            root=root->left;
            free(ptr);
        }
        else
        {
            struct TreeNode *temp = findMin(root->right);
            root->val = temp->val;
            root->right = deleteNode(root->right, temp->val);
        }
    }
    return root;
}
```



Leetcode Program:
Find bottom left tree value

```
int findBottomLeftValue(struct TreeNode* root) {
    if (root == NULL) {
        return -1;
    }

    struct TreeNode** queue = (struct TreeNode**)malloc(pow(10,4) * sizeof(struct
TreeNode*));
    int front = 0, rear = 0;
    int leftmostValue = 0;

    queue[rear++] = root;

    while (front < rear) {
        int levelSize = rear - front;

        for (int i = 0; i < levelSize; i++) {
            struct TreeNode* currentNode = queue[front++];

            if (i == 0) {
                leftmostValue = currentNode->val;
            }

            if (currentNode->left) {
                queue[rear++] = currentNode->left;
            }

            if (currentNode->right) {
```

```
                queue[rear++] = currentNode->right;
            }
        }
    }

    free(queue);
    return leftmostValue;
}
```