VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
On

DATA STRUCTURES (23CS3PCDST)

Submitted by

Pranav R Hegde (1BM22CS202)

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 Pranav R Hegde **(1BM22CS202)**, who is a 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 202324. 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. Sneha S Bagalkot	**Dr. Jyothi S Nayak**
Assistant Professor	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

Index Sheet

Course outcomes:

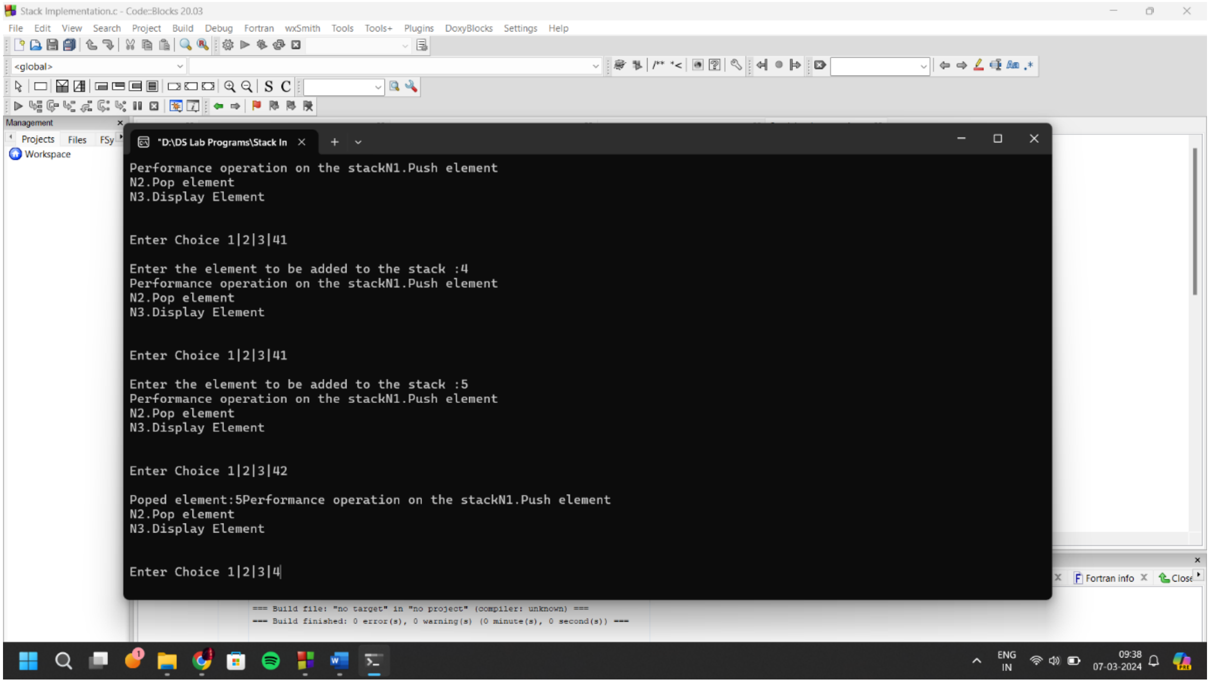
CO1	Apply the concept of linear and nonlinear data structures.
CO2	Analyze data structure operations for a given problem
CO3	Design and develop solutions using the operations of linear and nonlinear data structure for a given specification.
CO4	Conduct practical experiments for demonstrating the operations 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>
#define STACK_SIZE 5
void push(int st[],int *top)
{
    int item;
    if(*top==STACK_SIZE-1)
        printf("Stack overflow\n");
    else
    {
        printf("\nEnter an item :");
        scanf("%d",&item);
        (*top)++;
        st[*top]=item;
    }
}
void pop(int st[],int *top)
{ if(*top==-1) printf("Stack underflow\n");
    else
    {
        printf("\n%d item was deleted",st[(*top)--]);
    }
}
void display(int st[],int *top)
{
    int i;
    if(*top==-1) printf("Stack is
        empty\n");
    for(i=0;i<=*top;i++)
        printf("%d\t",st[i]);
} void
main()
{
    int st[10],top=-1, c,val_del;
    while(1)
    { printf("\n1. Push\n2. Pop\n3. Display\n");
        printf("\nEnter your choice :");
        scanf("%d",&c);
        switch(c)
        { case 1: push(st,&top);
            break;
```

```
                    case 2: pop(st,&top);
                            break;
                    case 3: display(st,&top);
                            break;
                    default: printf("\nInvalid choice!!!");
                            exit(0);
            }
        }
}
```



Stack Implementation.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSy
Workspace
"D:\DS Lab Programs\Stack In
Fortran info
Close
ENG IN
09:38 07-03-2024

```
Performance operation on the stackN1.Push element
N2.Pop element
N3.Display Element


Enter Choice 1|2|3|41

Enter the element to be added to the stack :4
Performance operation on the stackN1.Push element
N2.Pop element
N3.Display Element


Enter Choice 1|2|3|41

Enter the element to be added to the stack :5
Performance operation on the stackN1.Push element
N2.Pop element
N3.Display Element


Enter Choice 1|2|3|42

Poped element:5Performance operation on the stackN1.Push element
N2.Pop element
N3.Display Element


Enter Choice 1|2|3|4
```

```
=== Build file: "no target" in "no project" (compiler: unknown) ===
--- Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

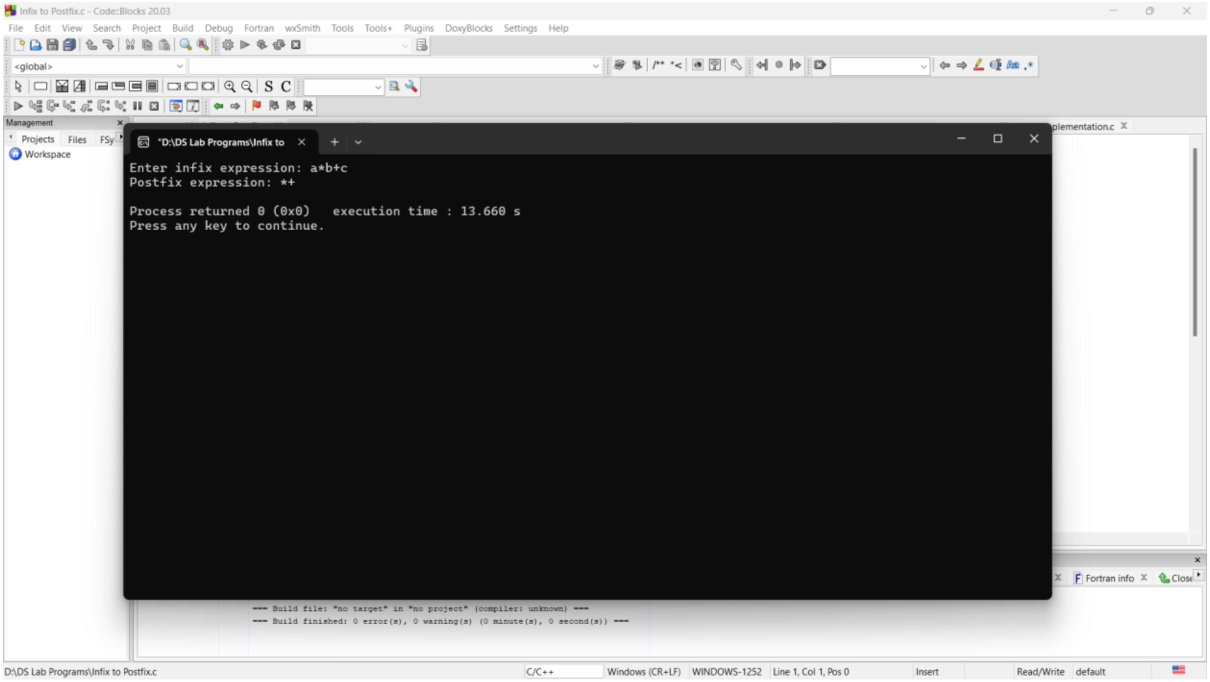
Lab Program 2:

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 <stdlib.h>




#define MAX_SIZE 100
```

```
int isOperator(char ch) {

    return (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '%');

}



int precedence(char operator) {

    if (operator == '+' || operator == '-')

        return 1;

    if (operator == '*' || operator == '/' || operator == '%')

        return 2;

    return 0;

}



void infixToPostfix(char infix[], char postfix[]) {

    char stack[MAX_SIZE];

    int top = -1;

    int i, j;
```

```
for (i = 0, j = 0; infix[i] != '\0'; i++) {
    if (infix[i] >= '0' && infix[i] <= '9') {
        postfix[j++] = infix[i];
    } else if (isOperator(infix[i])) {
        while (top >= 0 && precedence(stack[top]) >= precedence(infix[i])) {
            postfix[j++] = stack[top--];
        }
        stack[++top] = infix[i];
    } else if (infix[i] == '(') {
        stack[++top] = infix[i];
    } else if (infix[i] == ')') {
        while (top >= 0 && stack[top] != '(') {
            postfix[j++] = stack[top--];
        }
        if (top >= 0 && stack[top] == '(') {
            top--;
            }
        }
    }

    while (top >= 0) {
        postfix[j++] = stack[top--];
    }

    postfix[j] = '\0';
}
int main() {
    char infix[MAX_SIZE], postfix[MAX_SIZE];

    printf("Enter infix expression: ");
    scanf("%s", infix);

    infixToPostfix(infix, postfix);
    printf("Postfix expression: %s\n", postfix);



    return 0;

}
```


Infix to Postfix.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSy
Workspace
"D:\DS Lab Programs\Infix to
plementation.c
Fortran info
Close
D:\DS Lab Programs\Infix to Postfix.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

```
Enter infix expression: a*b+c
Postfix expression: *+

Process returned 0 (0x0)   execution time : 13.660 s
Press any key to continue.
```

```
=== Build file: "no target" in "no project" (compiler: unknown) ===
=== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

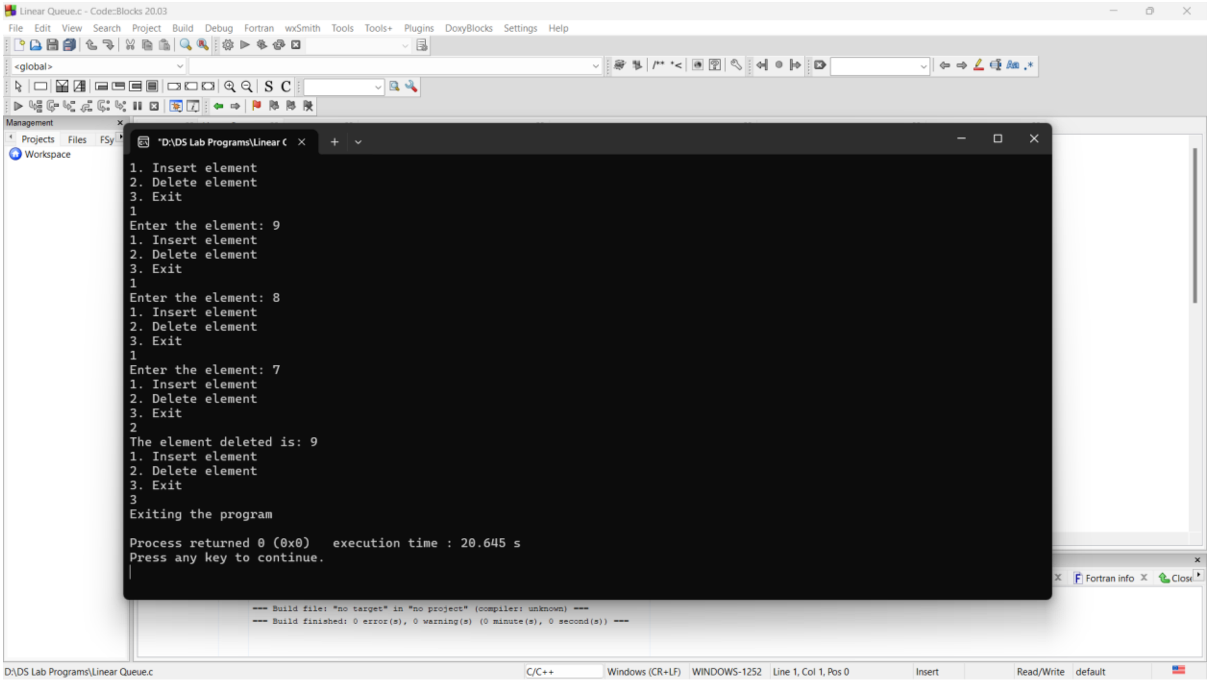
Lab Program 3:

3a) WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display The program should print appropriate messages for queue empty and queue overflow conditions

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

#define MAX 3

int front = -1, rear = -1;
int Q[MAX];

void insert(int item)
{
    if (rear == MAX - 1)
    {
        printf("Overflow\n");
        exit(EXIT_FAILURE);
    } else
    {
        if (rear == -1 && front == -1)
    {
      front = rear = 0;
      Q[rear] = item;
    }
  else
    {
      rear = rear + 1;
      Q[rear] = item;
    }
  }
}
int delete ()
{
  if (front == -1 || front > rear)
  {
    return -1;
  }
```

```
    else

    {

        return Q[front++];

    }

}

int main()

{

    int n,ele,d;

    do

    {

       printf("1. Insert element\n2. Delete element\n3. Exit\n");

        scanf("%d", &n);



        switch (n)

        {

            case 1:

                printf("Enter the element: ");
                scanf("%d", &ele);

                insert(ele);

                break;



            case 2:

                d = delete();

                if (d == -1)

                {

                    printf("Underflow\n");

                    exit(EXIT_FAILURE);

                }

                printf("The element deleted is: %d\n", d);

                break;

            case 3:

                printf("Exiting the program\n");

                break;

            default:
```

```
            printf("Please enter the right choice\n");

        }

    } while (n != 3);

    return 0;

}
```


Linear Queue.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSy
Workspace
"D:\DS Lab Programs\Linear C
Fortran info
Close
D:\DS Lab Programs\Linear Queue.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

```
1. Insert element
2. Delete element
3. Exit
1
Enter the element: 9
1. Insert element
2. Delete element
3. Exit
1
Enter the element: 8
1. Insert element
2. Delete element
3. Exit
1
Enter the element: 7
1. Insert element
2. Delete element
3. Exit
2
The element deleted is: 9
1. Insert element
2. Delete element
3. Exit
3
Exiting the program

Process returned 0 (0x0)   execution time : 20.645 s
Press any key to continue.
```

```
--- Build file: "no target" in "no project" (compiler: unknown) ---
--- Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

3b) WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display The program should print appropriate messages for queue empty and queue overflow conditions

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

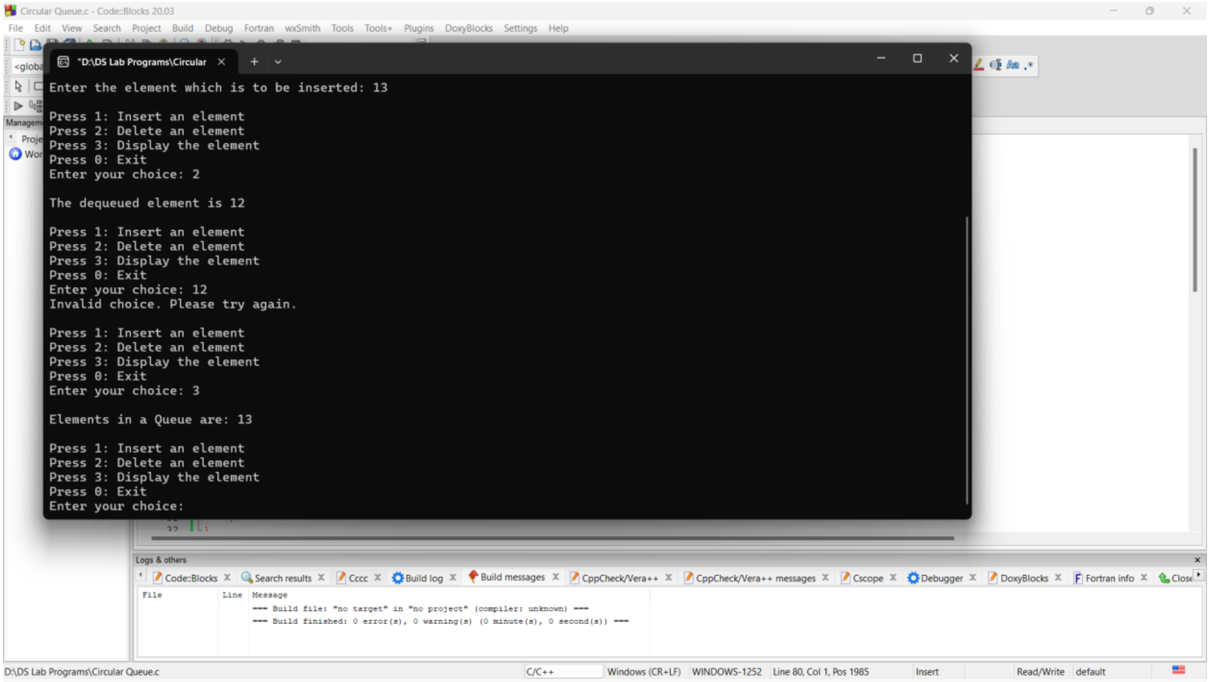
```
# define SIZE 5

int CQ[SIZE];

int front=-1;

int rear=-1;

void enqueue(int element)

{

   if(front==-1 && rear==-1)

   {

      front=0;

      rear=0;

      CQ[rear]=element;

   }

   else if((rear+1)%SIZE==front)

   {

      printf("Queue is overflow..");

   }

   else
```

```
    {
        rear=(rear+1)%SIZE;
        CQ[rear]=element;
    }
}
int dequeue()
{
    if((front==-1) && (rear==-1))
    {
        printf("\nQueue is underflow..");
    }
 else if(front==rear)
{
   printf("\nThe dequeued element is %d",CQ[front]);
   front=-1;
   rear=-1;
}
```

```
else

{

    printf("\nThe dequeued element is %d", CQ[front]);

   front=(front+1)%SIZE;

}

}

void display()

{

    int i=front;

    if(front==-1 && rear==-1)

    {

        printf("\n Queue is empty..");

    }

    else

    {

        printf("\nElements in a Queue are :");

        while(i<=rear)
        {

            printf("%d,", CQ[i]);

            i=(i+1)%SIZE;

        }

    }

}

int main()

{

    int choice=1,x;



    while(choice<4 && choice!=0)

    {

    printf("\n Press 1: Insert an element");

    printf("\nPress 2: Delete an element");

    printf("\nPress 3: Display the element");



    printf("\nEnter your choice");
```

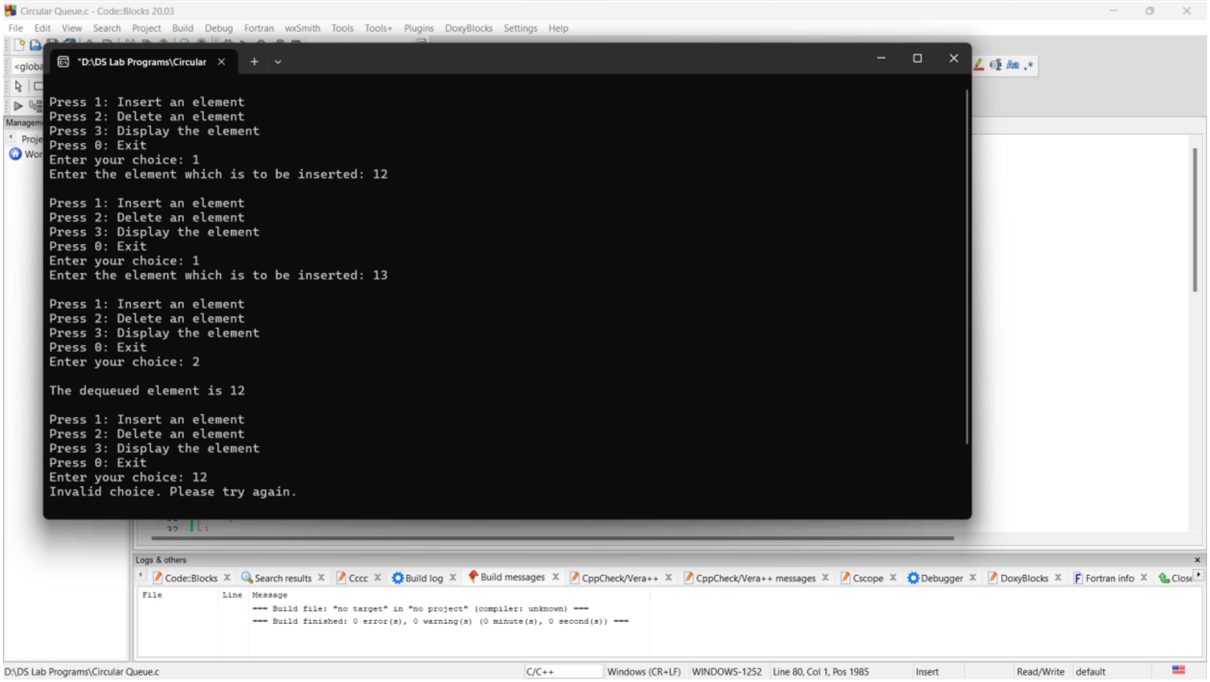
```
scanf("%d", &choice);

switch(choice)

{

   case 1:



   printf("Enter the element which is to be inserted");

   scanf("%d", &x);

   enqueue(x);

   break;

   case 2:

   dequeue();

   break;

   case 3:

   display();

}

}

return 0;
```

}



```
Enter the element which is to be inserted: 13

Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Press 0: Exit
Enter your choice: 2

The dequeued element is 12

Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Press 0: Exit
Enter your choice: 12
Invalid choice. Please try again.

Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Press 0: Exit
Enter your choice: 3

Elements in a Queue are: 13

Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Press 0: Exit
Enter your choice:
```

```
File        Line  Message
                  --- Build file: "no target" in "no project" (compiler: unknown) ---
                  --- Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```



```
Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Press 0: Exit
Enter your choice: 1
Enter the element which is to be inserted: 12

Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Press 0: Exit
Enter your choice: 1
Enter the element which is to be inserted: 13

Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Press 0: Exit
Enter your choice: 2

The dequeued element is 12

Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Press 0: Exit
Enter your choice: 12
Invalid choice. Please try again.
```

```
File        Line  Message
                  --- Build file: "no target" in "no project" (compiler: unknown) ---
                  --- Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

Lab Program 4:

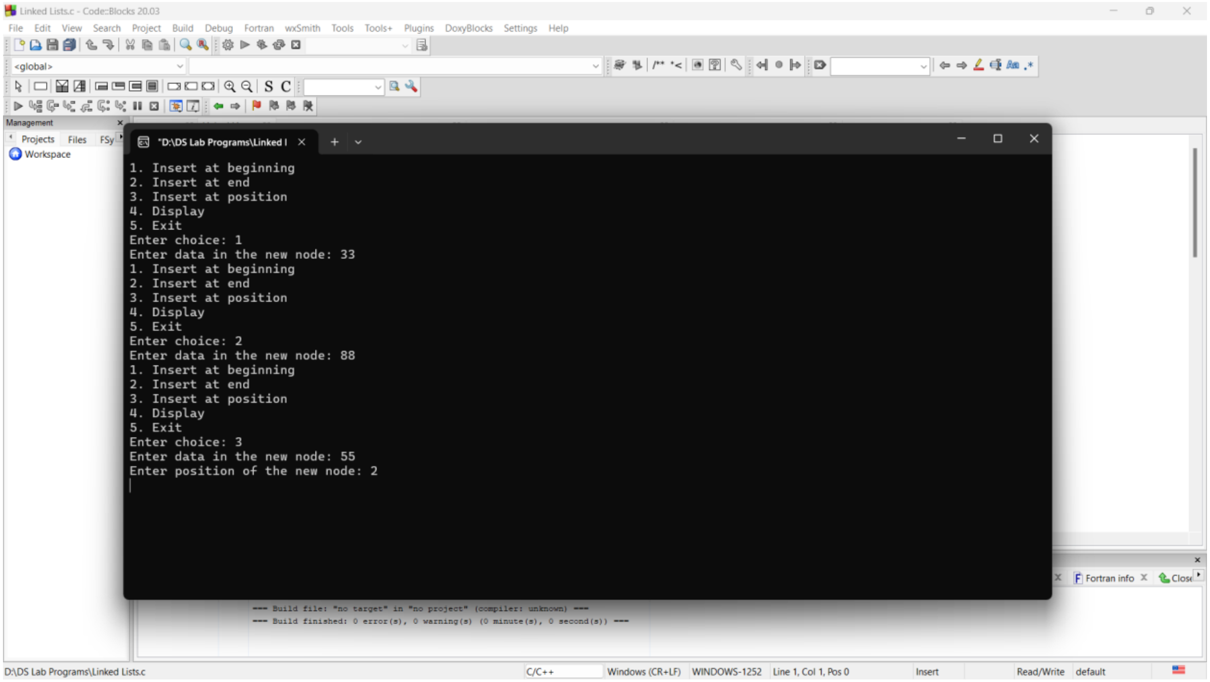
WAP to Implement Singly Linked List with following operations

a) Create a linked list.

b) Insertion of a node at first position, at any position and at end of list. Display the contents of the linked list.

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

typedef struct Node {
    int data;
    struct Node* next;
} Node;

Node* head = NULL;

void push();
void append();
void insert();
void display();

int main() {
    int choice;
    while (1) {
        printf("1. Insert at beginning\n");
        printf("2. Insert at end\n");
        printf("3. Insert at position\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                push();
                break;
            case 2:
                append();
                break;
            case 3:
                insert();
                break;
            case 4:
                display();
                break;
            default:
                printf("Exiting the program");
                return 0;
        }
    }
```

```
}

void push() {
   Node* temp = (Node*)malloc(sizeof(Node));
   int new_data;
   printf("Enter data in the new node: ");
   scanf("%d", &new_data);
   temp->data = new_data;
   temp->next = head;
   head = temp;
}

void append() {
   Node* temp = (Node*)malloc(sizeof(Node));
   int new_data;
   printf("Enter data in the new node: ");
   scanf("%d", &new_data);
   temp->data = new_data;
   temp->next = NULL;
   if (head == NULL) {
      head = temp;
      return;
   }
   Node* temp1 = head;
   while (temp1->next != NULL) {
      temp1 = temp1->next;
   }
   temp1->next = temp;
}

void insert() {
   Node* temp = (Node*)malloc(sizeof(Node));
   int new_data, pos;
   printf("Enter data in the new node: ");
   scanf("%d", &new_data);
   printf("Enter position of the new node: ");
   scanf("%d", &pos);
   temp->data = new_data;
   temp->next = NULL;
   if (pos == 0) {
      temp->next = head;
      head = temp;
      return;
   }
   Node* temp1 = head;
   while (pos--) {
        temp1 = temp1->next;
    }
    Node* temp2 = temp1->next;
    temp->next = temp2;
    temp1->next = temp;
}

void display() {
    Node* temp1 = head;
    while (temp1 != NULL) {
        printf("%d -> ", temp1->data);
        temp1 = temp1->next;
    }
    printf("NULL\n");
}
```


Linked Lists.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSy
Workspace
"D:\DS Lab Programs\Linked I
Fortran info
Close
D:\DS Lab Programs\Linked Lists.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

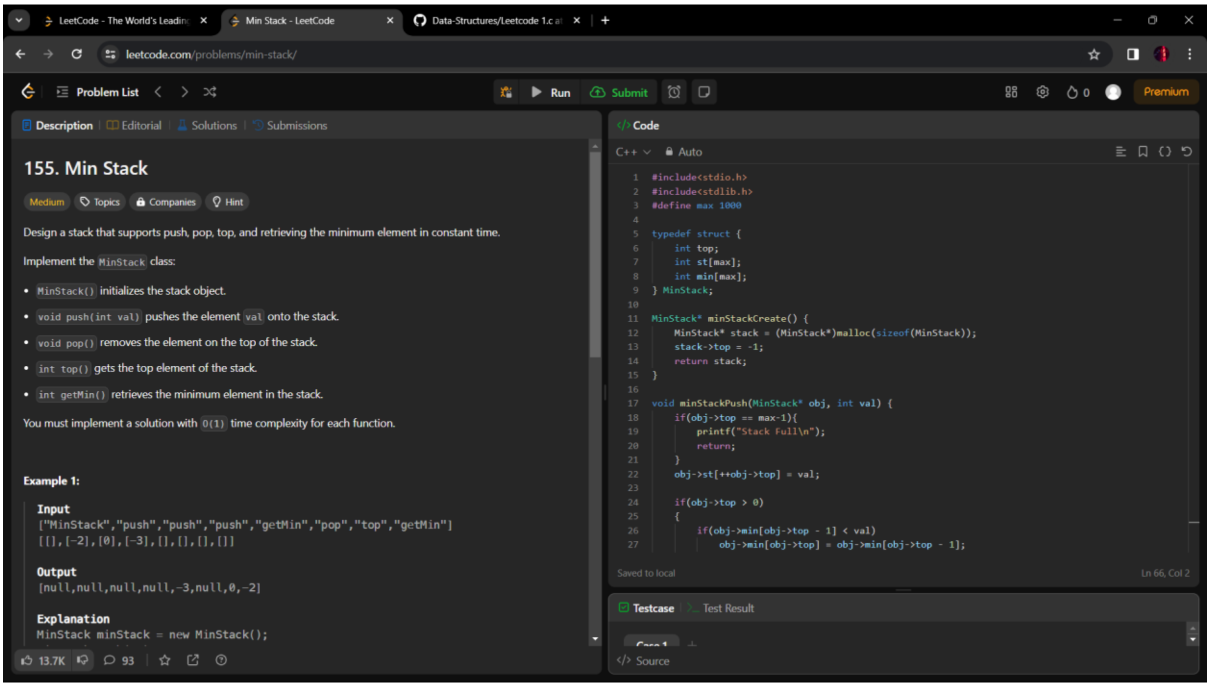
```
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter choice: 1
Enter data in the new node: 33
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter choice: 2
Enter data in the new node: 88
1. Insert at beginning
2. Insert at end
3. Insert at position
4. Display
5. Exit
Enter choice: 3
Enter data in the new node: 55
Enter position of the new node: 2
```

```
--- Build file: "no target" in "no project" (compiler: unknown) ---
--- Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

Leetcode Program: Min Stack

```
#include<stdio.h>
#include<stdlib.h>
#define max 1000

typedef struct {
    int top;
    int st[max];
    int min[max];
} MinStack;

MinStack* minStackCreate() {
    MinStack* stack = (MinStack*)malloc(sizeof(MinStack));
    stack->top = -1;
    return stack;
}

void minStackPush(MinStack* obj, int val) {
    if(obj->top == max-1){
       printf("Stack Full\n");
       return;
    }
    obj->st[++obj->top] = val;

    if(obj->top > 0)
    {
       if(obj->min[obj->top - 1] < val)
          obj->min[obj->top] = obj->min[obj->top - 1];
       else
          obj->min[obj->top] = val;
    }
    else
       obj->min[obj->top] = val;
}

void minStackPop(MinStack* obj) {
    if(obj->top == -1)
    {
       printf("Stack empty\n");
       return;
    }
    else {
       obj->top -= 1;
    }
}

int minStackTop(MinStack* obj) {
    if(obj->top == -1)
    {
       printf("Stack empty\n");
       return -1;
    }
    return obj->st[obj->top];
}
```

```
int minStackGetMin(MinStack* obj) {
    if(obj->top == -1)
    {
        printf("min Stack empty\n");
        return -1;
    }
    return obj->min[obj->top];
}

void minStackFree(MinStack* obj) {
    free(obj);
}
```



155. Min Stack

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the `MinStack` class:

- `MinStack()` initializes the stack object.
- `void push(int val)` pushes the element `val` onto the stack.
- `void pop()` removes the element on the top of the stack.
- `int top()` gets the top element of the stack.
- `int getMin()` retrieves the minimum element in the stack.

You must implement a solution with `O(1)` time complexity for each function.

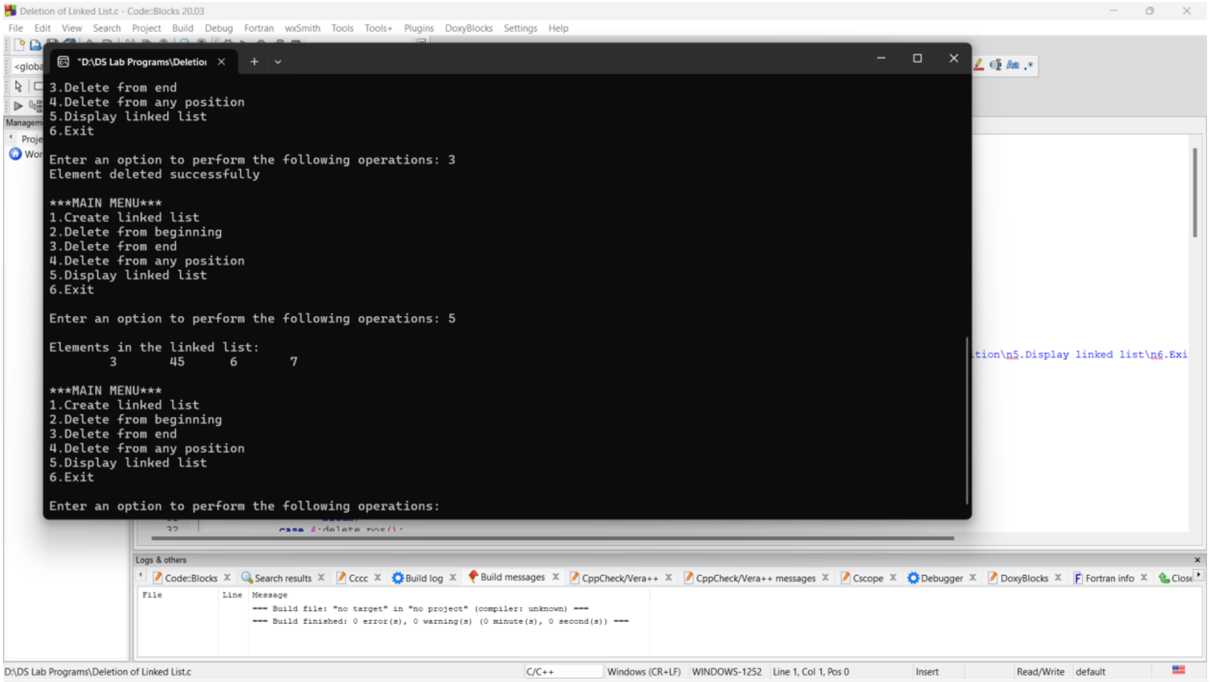
Example 1:

```
Input
["MinStack","push","push","push","getMin","pop","top","getMin"]
[[],[-2],[0],[-3],[],[],[],[]]

Output
[null,null,null,null,-3,null,0,-2]

Explanation
MinStack minStack = new MinStack();
```

```
#include<stdio.h>
#include<stdlib.h>
#define max 1000

typedef struct {
    int top;
    int st[max];
    int min[max];
} MinStack;

MinStack* minStackCreate() {
    MinStack* stack = (MinStack*)malloc(sizeof(MinStack));
    stack->top = -1;
    return stack;
}

void minStackPush(MinStack* obj, int val) {
    if(obj->top == max-1){
        printf("Stack Full\n");
        return;
    }
    obj->st[++obj->top] = val;

    if(obj->top > 0)
    {
        if(obj->min[obj->top - 1] < val)
            obj->min[obj->top] = obj->min[obj->top - 1];
```

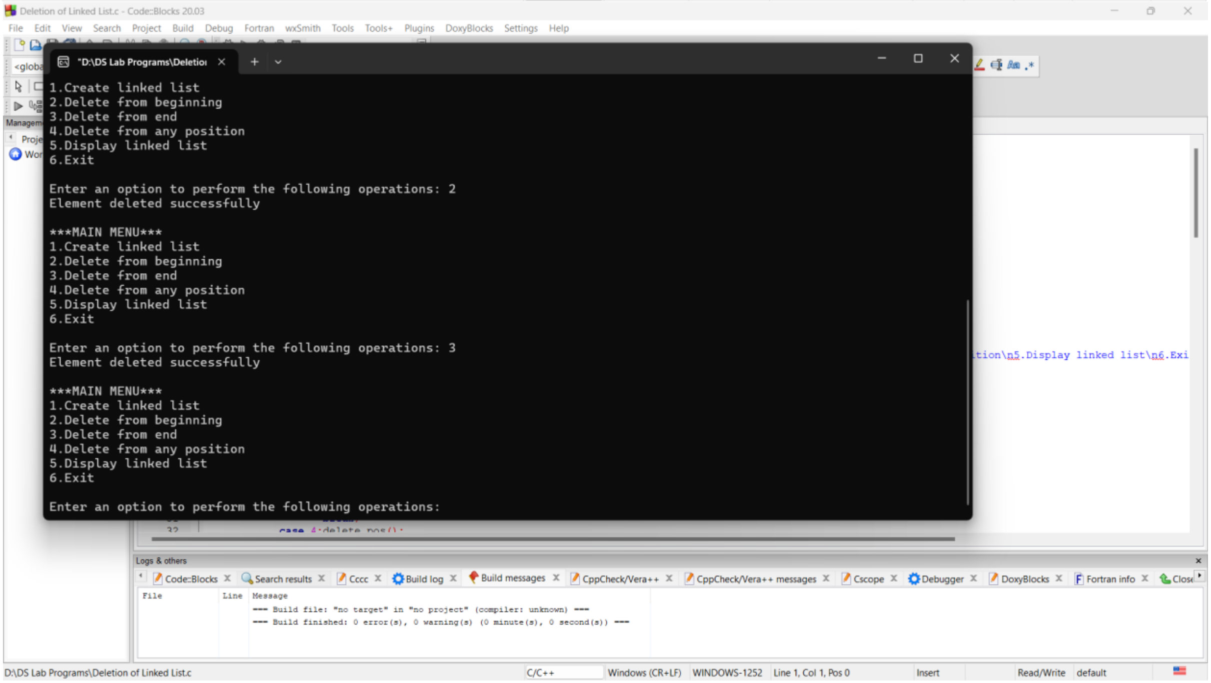
Lab Program 5:
WAP to Implement Singly Linked List with following operations
a) Create a linked list.
b) Deletion of first element, specified element and last element in the list.
c) Display the contents of the linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *start=NULL;
void create();
void delete_begin();
void delete_end();
void delete_pos();
void display();
int main()
{
    int option;
    do{
        printf("\n***MAIN MENU***\n1.Create linked list\n2.Delete from
beginning\n3.Delete from end\n4.Delete from any position\n5.Display linked
list\n6.Exit\n");
        printf("\nEnter an option to perform the following operations: ");
        scanf("%d",&option);
        switch(option)
        {
            case 1:create();
                   printf("\nLinked list created successfully\n");
                   break;
            case 2:delete_begin();
                   printf("Element deleted successfully\n");
                   break;
            case 3:delete_end();
                   printf("Element deleted successfully\n");
                   break;
            case 4:delete_pos();
                   printf("Element deleted successfully\n");
                   break;
            case 5:printf("\nElements in the linked list:\n");
                   display();
                   break;
        }
    }while(option!=6);
    return 0;
}
void create()
{
    struct node *ptr,*new_node;
    int num;
    printf("Enter -1 to exit\n");
    printf("\nEnter the data\n");
    scanf("%d",&num);
    while(num!=-1)
    {
        new_node=(struct node*)malloc(sizeof(struct node));
        new_node->data=num;
        if(start==NULL)
        {
            start=new_node;
            new_node->next=NULL;
        }
        else
        {
            ptr=start;
            while(ptr->next!=NULL)
            ptr=ptr->next;
            ptr->next=new_node;
            new_node->next=NULL;
        }
        printf("Enter the data\n");
        scanf("%d",&num);
    }
}
void delete_begin()
{
    struct node *ptr;
    ptr=start;
    start=start->next;
    free(ptr);
}
void delete_end()
{
    struct node *ptr,*preptr;
    ptr=start;
    while(ptr->next!=NULL)
    {
    preptr=ptr;
    ptr=ptr->next;
    }
    preptr->next=NULL;
    free(ptr);
}
void delete_pos()
{
    struct node *ptr,*preptr,*postptr;
    int pos,count=1;
    printf("Enter the position: ");
    scanf("%d",&pos);
    ptr=start;
    if(pos==1)
    {
        start=start->next;
        free(ptr);
    }
    else
    {
        while(count<pos&&ptr!=NULL)
        {
            preptr=ptr;
            ptr=ptr->next;
            postptr=ptr->next;
            count++;
        }
        if(pos==count)
        {
            preptr->next=postptr;
            free(ptr);
        }

    }
}
void display()
{
    struct node *ptr;
    ptr=start;
    while(ptr!=NULL)
    {
        printf("\t%d",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
```


Deletion of Linked List.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\DS Lab Programs\Deletio
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close
D:\DS Lab Programs\Deletion of Linked List.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

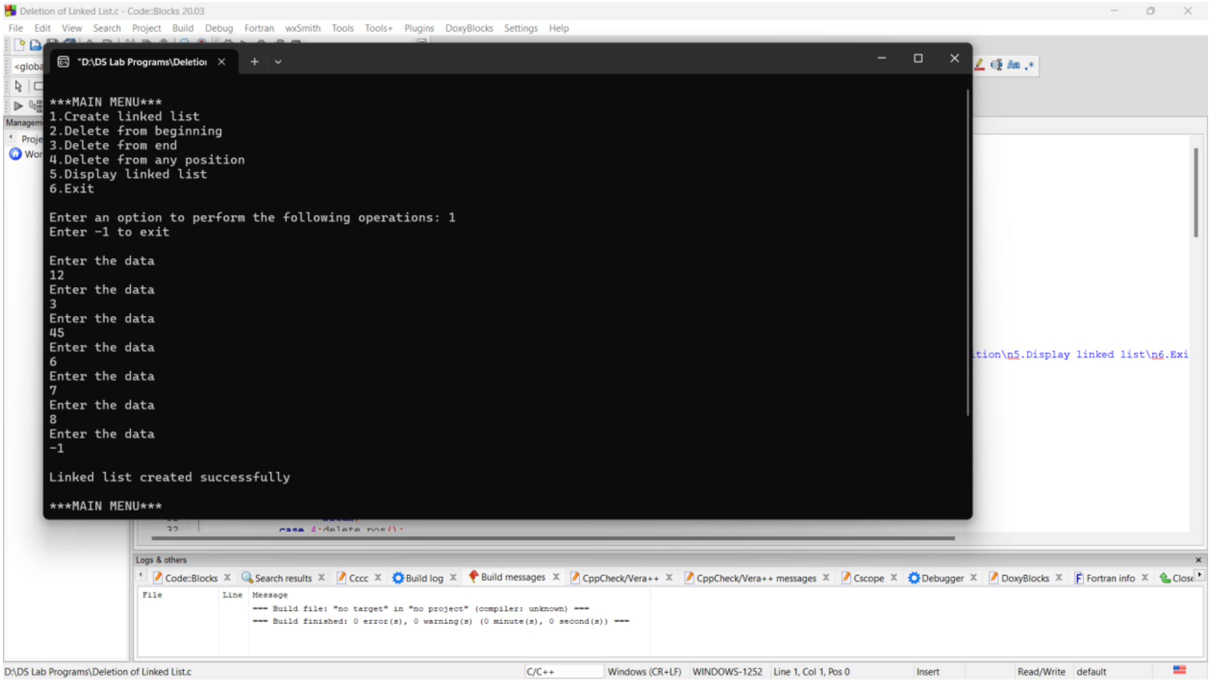
```
3.Delete from end
4.Delete from any position
5.Display linked list
6.Exit

Enter an option to perform the following operations: 3
Element deleted successfully

***MAIN MENU***
1.Create linked list
2.Delete from beginning
3.Delete from end
4.Delete from any position
5.Display linked list
6.Exit

Enter an option to perform the following operations: 5

Elements in the linked list:
        3       45      6       7

***MAIN MENU***
1.Create linked list
2.Delete from beginning
3.Delete from end
4.Delete from any position
5.Display linked list
6.Exit

Enter an option to perform the following operations:
```

```
File    Line  Message
              === Build file: "no target" in "no project" (compiler: unknown) ===
              === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```


Deletion of Linked List.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\DS Lab Programs\Deletio
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close
D:\DS Lab Programs\Deletion of Linked List.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

```
1.Create linked list
2.Delete from beginning
3.Delete from end
4.Delete from any position
5.Display linked list
6.Exit

Enter an option to perform the following operations: 2
Element deleted successfully

***MAIN MENU***
1.Create linked list
2.Delete from beginning
3.Delete from end
4.Delete from any position
5.Display linked list
6.Exit

Enter an option to perform the following operations: 3
Element deleted successfully

***MAIN MENU***
1.Create linked list
2.Delete from beginning
3.Delete from end
4.Delete from any position
5.Display linked list
6.Exit

Enter an option to perform the following operations:
```

```
File    Line  Message
              === Build file: "no target" in "no project" (compiler: unknown) ===
              === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```


Deletion of Linked List.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\DS Lab Programs\Deletio
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks
Fortran info
D:\DS Lab Programs\Deletion of Linked List.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

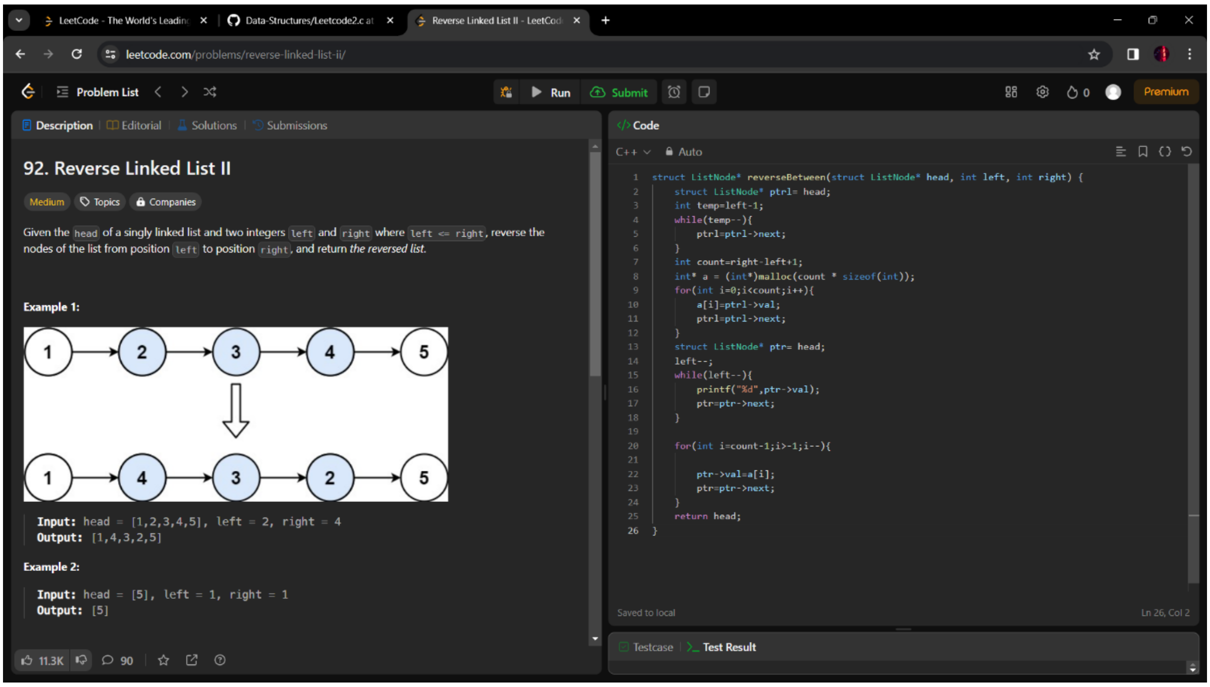
```
***MAIN MENU***
1.Create linked list
2.Delete from beginning
3.Delete from end
4.Delete from any position
5.Display linked list
6.Exit

Enter an option to perform the following operations: 1
Enter -1 to exit

Enter the data
12
Enter the data
3
Enter the data
45
Enter the data
6
Enter the data
7
Enter the data
8
Enter the data
-1

Linked list created successfully

***MAIN MENU***
```

```
File      Line  Message
                === Build file: "no target" in "no project" (compiler: unknown) ===
                === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

Leetcode Program: Reverse Linked List

```
struct ListNode* reverseBetween(struct ListNode* head, int left, int right) {
    struct ListNode* ptrl= head;
    int temp=left-1;
    while(temp--){
        ptrl=ptrl->next;
    }
    int count=right-left+1;
    int* a = (int*)malloc(count * sizeof(int));
    for(int i=0;i<count;i++){
        a[i]=ptrl->val;
        ptrl=ptrl->next;
    }
    struct ListNode* ptr= head;
    left--;
    while(left--){
        printf("%d",ptr->val);
        ptr=ptr->next;
    }

    for(int i=count-1;i>-1;i--){

        ptr->val=a[i];
        ptr=ptr->next;
    }
    return head;
```

}



92. Reverse Linked List II

Given the `head` of a singly linked list and two integers `left` and `right` where `left <= right`, reverse the nodes of the list from position `left` to position `right`, and return *the reversed list*.

Example 1:

```
Input: head = [1,2,3,4,5], left = 2, right = 4
Output: [1,4,3,2,5]
```

Example 2:

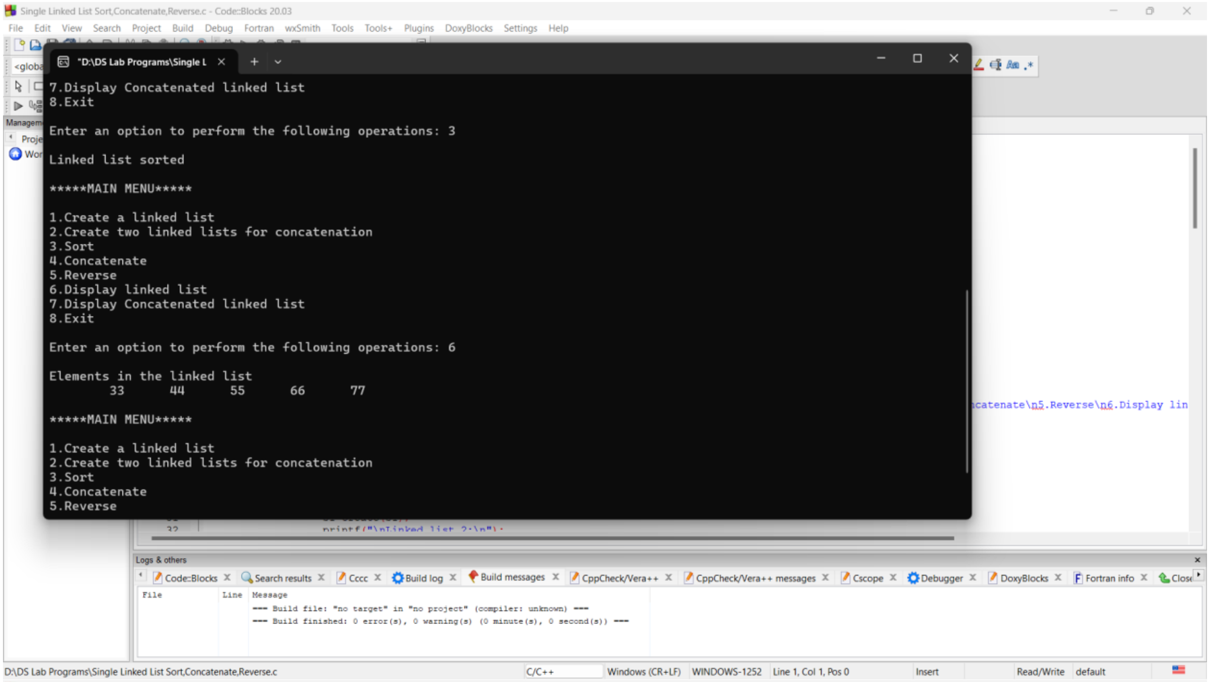
```
Input: head = [5], left = 1, right = 1
Output: [5]
```

```
struct ListNode* reverseBetween(struct ListNode* head, int left, int right) {
    struct ListNode* ptrl= head;
    int temp=left-1;
    while(temp--){
        ptrl=ptrl->next;
    }
    int count=right-left+1;
    int* a = (int*)malloc(count * sizeof(int));
    for(int i=0;i<count;i++){
        a[i]=ptrl->val;
        ptrl=ptrl->next;
    }
    struct ListNode* ptr= head;
    left--;
    while(left--){
        printf("%d",ptr->val);
        ptr=ptr->next;
    }

    for(int i=count-1;i>-1;i--){

        ptr->val=a[i];
        ptr=ptr->next;
    }
    return head;
}
```

Lab Program 6:

6a) WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.

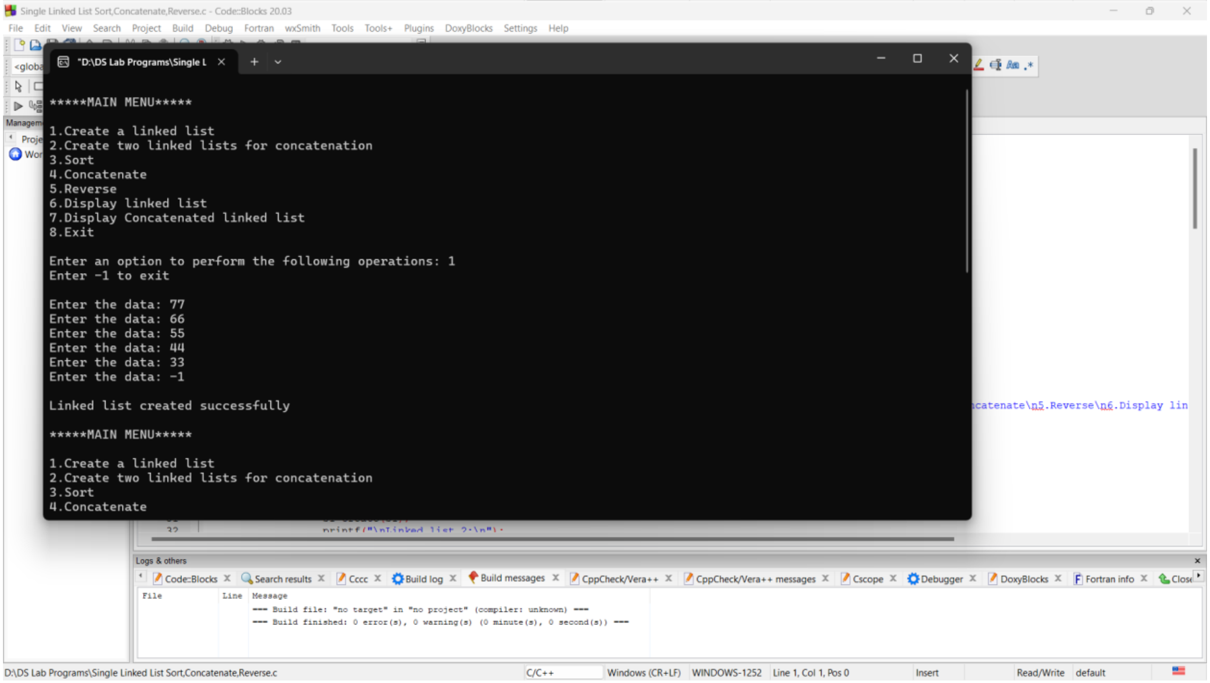
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *s1=NULL;
struct node *s2=NULL;
struct node *start=NULL;
struct node *create(struct node*);
void sort();
struct node *concatenate(struct node*,struct node*);
void reverse();
void display(struct node*);

int main()
{
   int option;
   struct node *a=NULL;
   do{
        printf("\n*****MAIN MENU*****\n\n1.Create a linked list\n2.Create two linked
lists for concatenation\n3.Sort\n4.Concatenate\n5.Reverse\n6.Display linked
list\n7.Display Concatenated linked list\n8.Exit\n");
        printf("\nEnter an option to perform the following operations: ");
        scanf("%d",&option);
        switch(option)
        {
          case 1:start=create(start);
                printf("\nLinked list created successfully\n");
                break;
          case 2:printf("\nLinked list 1:\n");
                s1=create(s1);
                printf("\nLinked list 2:\n");
                s2=create(s2);
                printf("\nLinked lists created successfully\n");
                break;
          case 3:sort();
                printf("\nLinked list sorted\n");
                break;
          case 4:a=concatenate(s1,s2);
                printf("\nLinked lists concatenated successfully\n");
                break;
          case 5:reverse();
                printf("\nLinked list reversed\n");
                break;
          case 6:printf("\nElements in the linked list\n");
                display(start);
                break;
          case 7:printf("\nElements in the linked list after concatenation:\n");
                display(a);
                break;

        }
    }while(option!=8);
    return 0;
}

struct node * create(struct node *start)
{
    struct node *ptr,*new_node;
    int num;
    printf("Enter -1 to exit\n");
    printf("\nEnter the data: ");
    scanf("%d",&num);
    while(num!=-1)
    {
        new_node=(struct node*)malloc(sizeof(struct node));
        new_node->data=num;
        if(start==NULL)
        {
            start=new_node;
            new_node->next=NULL;
        }
        else
        {
            ptr=start;
            while(ptr->next!=NULL)
            ptr=ptr->next;
            ptr->next=new_node;
            new_node->next=NULL;
        }
        printf("Enter the data: ");
        scanf("%d",&num);
    }
    return start;
}

void sort()
{
    struct node *i,*j;
    int temp;
    for(i=start;i->next!=NULL;i=i->next)
    {
        for(j=i->next;j!=NULL;j=j->next)
        {
          if(i->data>j->data)
          {
            temp=i->data;
            i->data=j->data;
            j->data=temp;
          }
        }
    }
}

struct node *concatenate(struct node *t1,struct node *t2)
{
    struct node *ptr;
    ptr=t1;
    while(ptr->next!=NULL)
    {
     ptr=ptr->next;
    }
ptr->next=t2;
return t1;
}

void reverse()
{
  struct node *prev=NULL;
  struct node *next=NULL;
  struct node *cur=start;
  while(cur!=NULL)
  {
    next=cur->next;
    cur->next=prev;
    prev=cur;
    cur=next;
  }
  start=prev;
}

void display(struct node *p)
{
    struct node *ptr;
    ptr=p;
    while(ptr!=NULL)
    {
        printf("\t%d",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
```



Single Linked List Sort,Concatenate,Reverse.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\DS Lab Programs\Single L
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close
D:\DS Lab Programs\Single Linked List Sort,Concatenate,Reverse.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

```
7.Display Concatenated linked list
8.Exit

Enter an option to perform the following operations: 3

Linked list sorted

*****MAIN MENU*****

1.Create a linked list
2.Create two linked lists for concatenation
3.Sort
4.Concatenate
5.Reverse
6.Display linked list
7.Display Concatenated linked list
8.Exit

Enter an option to perform the following operations: 6

Elements in the linked list
        33      44      55      66      77

*****MAIN MENU*****

1.Create a linked list
2.Create two linked lists for concatenation
3.Sort
4.Concatenate
5.Reverse
```

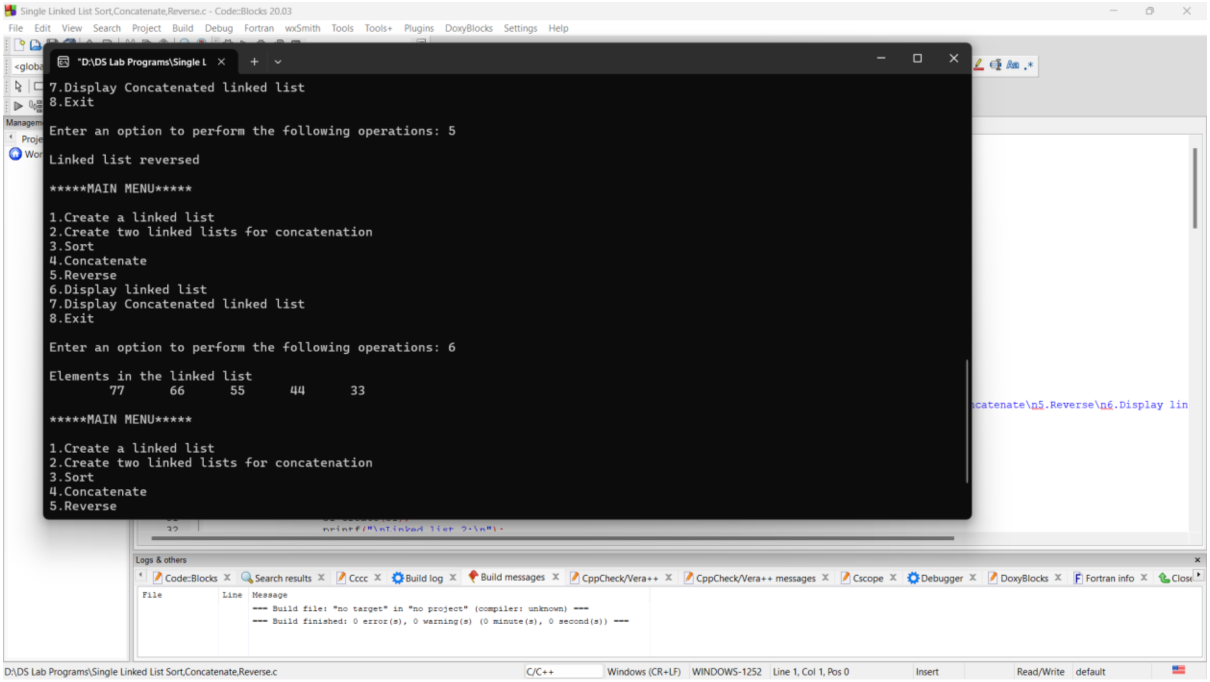
```
File    Line  Message
              === Build file: "no target" in "no project" (compiler: unknown) ===
              === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```



Single Linked List Sort,Concatenate,Reverse.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\DS Lab Programs\Single L
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close
D:\DS Lab Programs\Single Linked List Sort,Concatenate,Reverse.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

```
*****MAIN MENU*****

1.Create a linked list
2.Create two linked lists for concatenation
3.Sort
4.Concatenate
5.Reverse
6.Display linked list
7.Display Concatenated linked list
8.Exit

Enter an option to perform the following operations: 1
Enter -1 to exit

Enter the data: 77
Enter the data: 66
Enter the data: 55
Enter the data: 44
Enter the data: 33
Enter the data: -1

Linked list created successfully

*****MAIN MENU*****

1.Create a linked list
2.Create two linked lists for concatenation
3.Sort
4.Concatenate
```

```
File    Line  Message
              === Build file: "no target" in "no project" (compiler: unknown) ===
              === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```


Single Linked List Sort,Concatenate,Reverse.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\DS Lab Programs\Single L
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close
D:\DS Lab Programs\Single Linked List Sort,Concatenate,Reverse.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

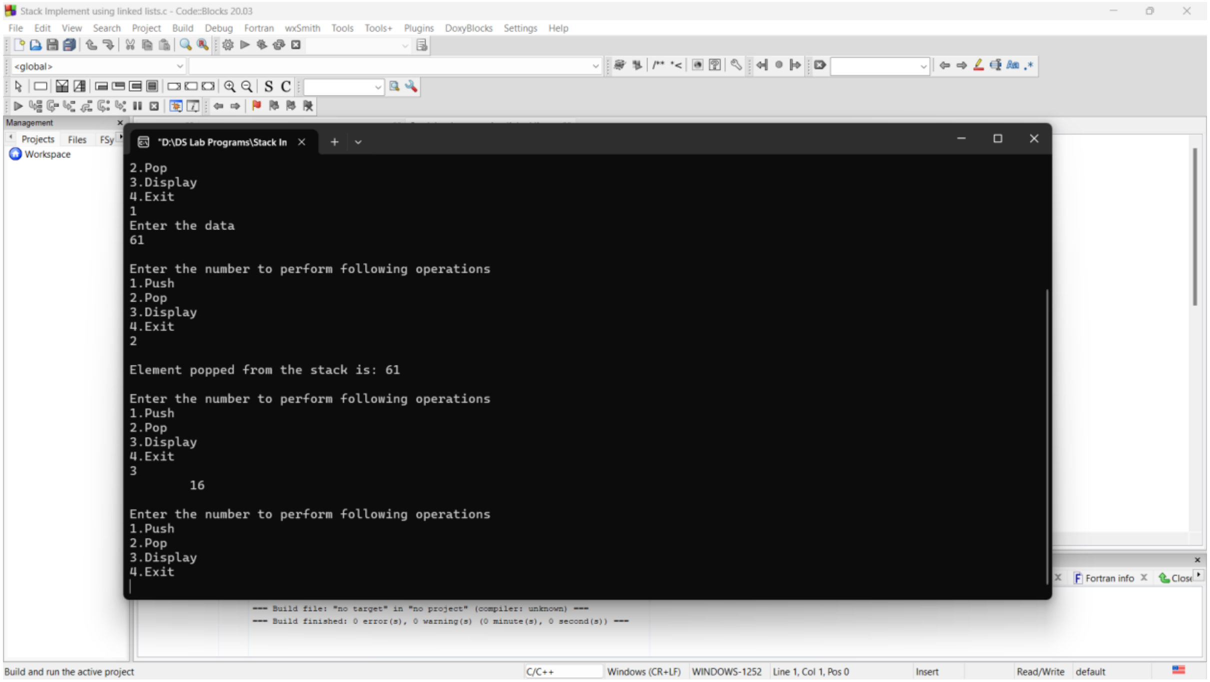
```
7.Display Concatenated linked list
8.Exit

Enter an option to perform the following operations: 5

Linked list reversed

*****MAIN MENU*****

1.Create a linked list
2.Create two linked lists for concatenation
3.Sort
4.Concatenate
5.Reverse
6.Display linked list
7.Display Concatenated linked list
8.Exit

Enter an option to perform the following operations: 6

Elements in the linked list
        77      66      55      44      33

*****MAIN MENU*****

1.Create a linked list
2.Create two linked lists for concatenation
3.Sort
4.Concatenate
5.Reverse
```

```
File          Line  Message
                    === Build file: "no target" in "no project" (compiler: unknown) ===
                    === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

6b) WAP to Implement Single Link List to simulate Stack & Queue Operations

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int data;
  struct node *next;
};
struct node *start=NULL;
void push();
void pop();
void display();
int main()
{
   int val,option;
    do
    {
        printf("\nEnter the number to perform following
operations\n1.Push\n2.Pop\n3.Display\n4.Exit\n");
        scanf("%d",&option);
        switch(option)
        {
            case 1:push();
            break;
            case 2:pop();
            break;
            case 3:display();
            break;
        }
    }
    while(option!=4);
    return 0;
}
void push()
{
    struct node *new_node;
    int num;
    printf("Enter the data\n");
    scanf("%d",&num);
    new_node=(struct node*)malloc(sizeof(struct node));
    new_node->data=num;
    new_node->next=start;
    start=new_node;
}
void pop()
{
  struct node *ptr;
  ptr=start;
  if(start==NULL)
  {
  printf("Stack is empty\n");
  exit(0);
  }
  else
  {
    ptr=start;
    start=ptr->next;
    printf("\nElement popped from the stack is: %d\n",ptr->data);
    free(ptr);
  }

}
void display()
{

    struct node *ptr;
    ptr=start;
    while(ptr!=NULL)
    {
        printf("\t%d",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
```

```
}
```



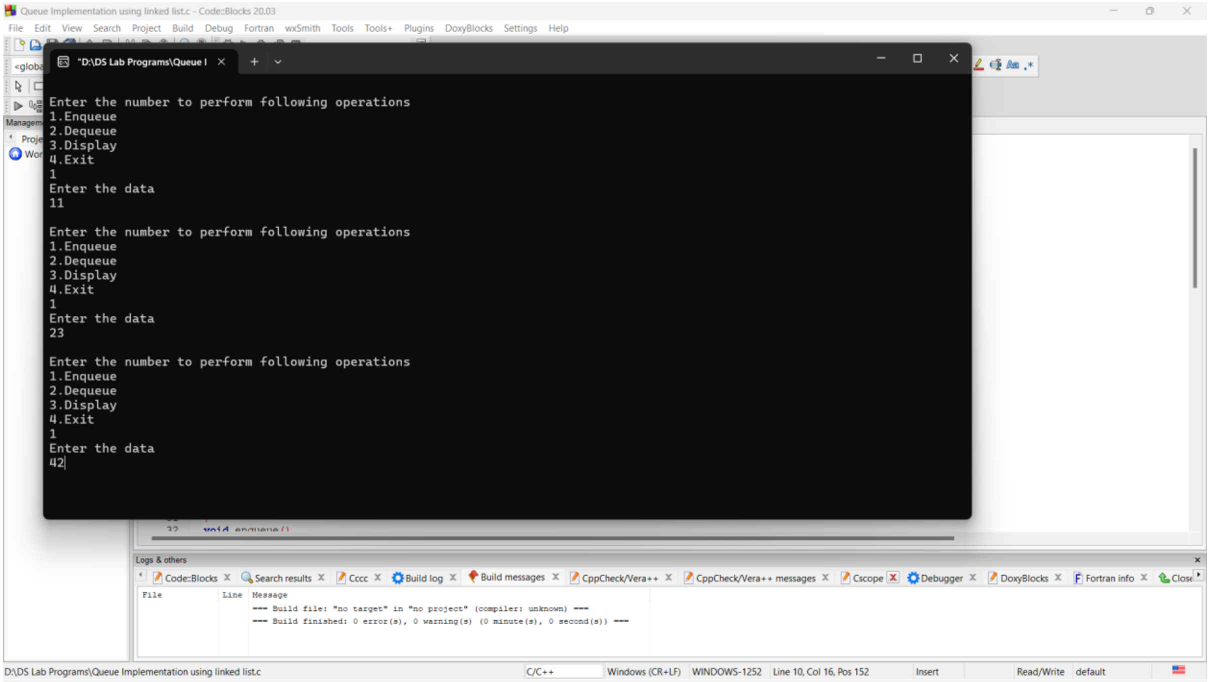
Stack Implement using linked lists.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSy
Workspace
*D:\DS Lab Programs\Stack In
Fortran info
Close
Build and run the active project
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

```
2.Pop
3.Display
4.Exit
1
Enter the data
61

Enter the number to perform following operations
1.Push
2.Pop
3.Display
4.Exit
2

Element popped from the stack is: 61

Enter the number to perform following operations
1.Push
2.Pop
3.Display
4.Exit
3
        16

Enter the number to perform following operations
1.Push
2.Pop
3.Display
4.Exit
```

```
--- Build file: "no target" in "no project" (compiler: unknown) ---
--- Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

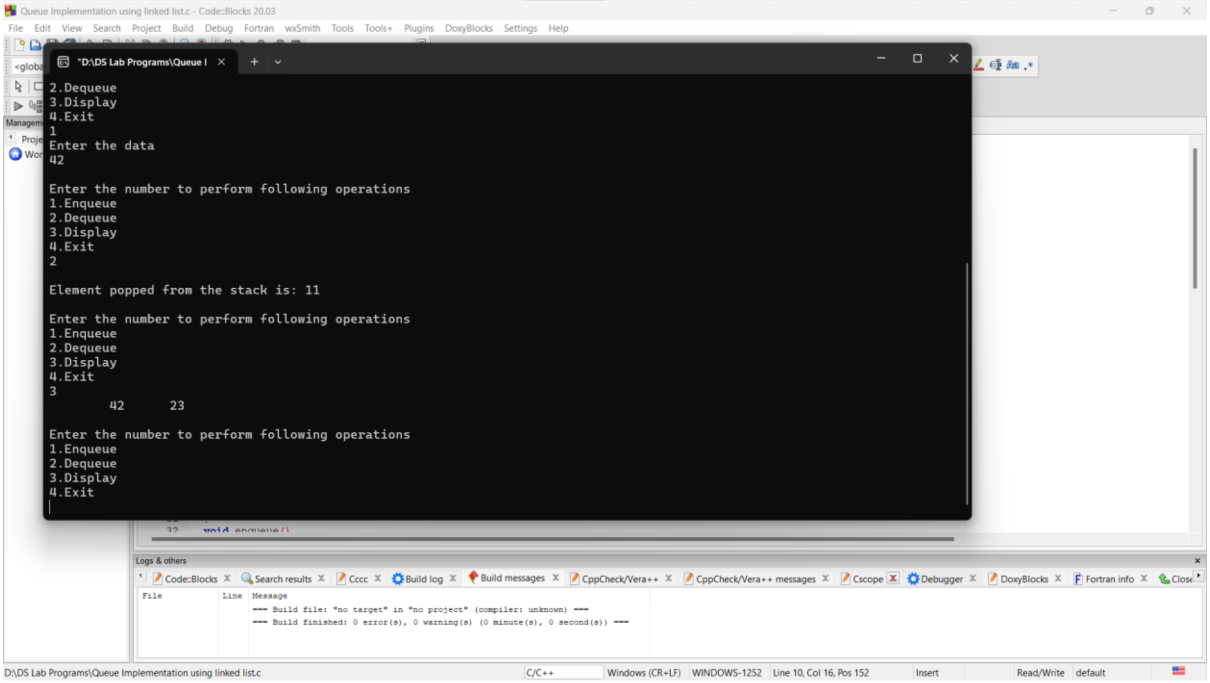
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int data;
  struct node *next;
};
struct node *start=NULL;
void enqueue();
void dequeue();
void display();
int main()
{
   int val,option;
    do
    {
        printf("\nEnter the number to perform following
operations\n1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\n");
        scanf("%d",&option);
        switch(option)
        {
            case 1:enqueue();
            break;
            case 2:dequeue();
            break;
            case 3:display();
            break;
        }
    }
    while(option!=4);
    return 0;
}
void enqueue()
{
    struct node *new_node;
    int num;
    printf("Enter the data\n");
    scanf("%d",&num);
    new_node=(struct node*)malloc(sizeof(struct node));
    new_node->data=num;
    new_node->next=start;
    start=new_node;
}
void dequeue()
{
  struct node *ptr,*preptr;
  ptr=start;
  if(start==NULL)
  {
  printf("Stack is empty\n");
  exit(0);
  }
  else if(start->next==NULL)
  {
    start=start->next;
    printf("\nElement popped from the stack is: %d\n",ptr->data);
    free(ptr);
  }
  else
  {
    while(ptr->next!=NULL)
    {
    preptr=ptr;
    ptr=ptr->next;
    }
    preptr->next=NULL;
    printf("\nElement popped from the stack is: %d\n",ptr->data);
    free(ptr);
  }
}
```

```
void display()
{

   struct node *ptr;
   ptr=start;
   while(ptr!=NULL)
   {
      printf("\t%d",ptr->data);
      ptr=ptr->next;
   }
   printf("\n");
}
```



```
Enter the number to perform following operations
1.Enqueue
2.Dequeue
3.Display
4.Exit
1
Enter the data
11

Enter the number to perform following operations
1.Enqueue
2.Dequeue
3.Display
4.Exit
1
Enter the data
23

Enter the number to perform following operations
1.Enqueue
2.Dequeue
3.Display
4.Exit
1
Enter the data
42
```

```
File    Line  Message
              === Build file: "no target" in "no project" (compiler: unknown) ===
              === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```



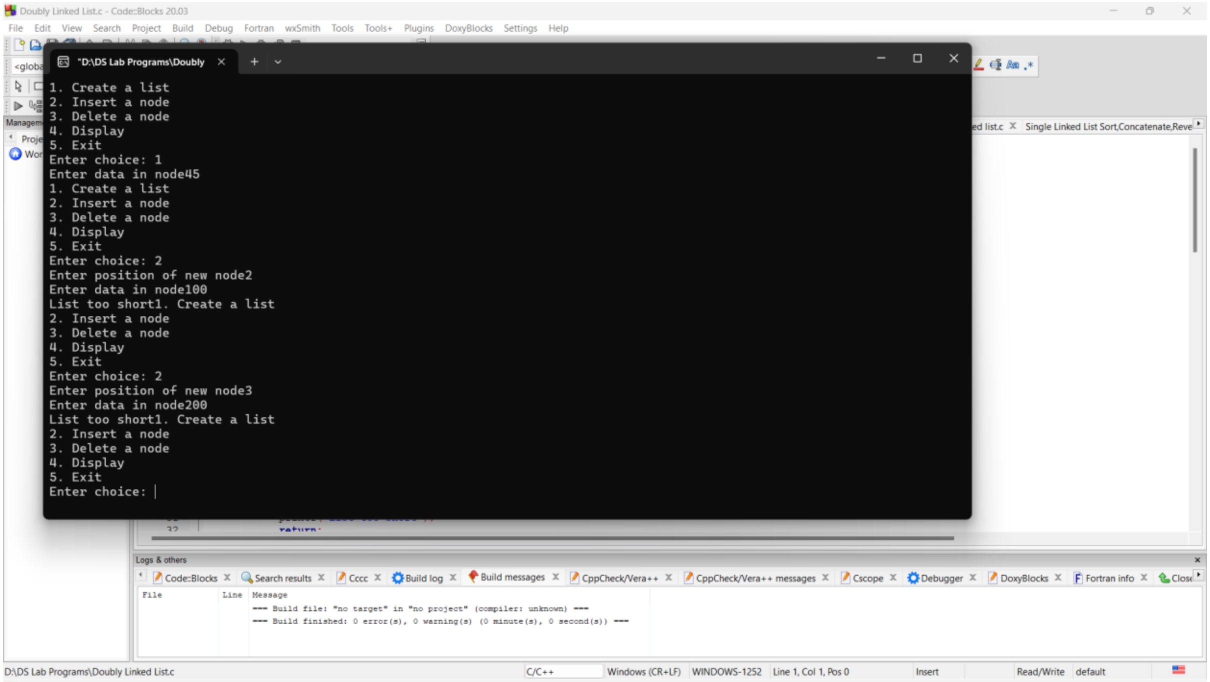
```
2.Dequeue
3.Display
4.Exit
1
Enter the data
42

Enter the number to perform following operations
1.Enqueue
2.Dequeue
3.Display
4.Exit
2

Element popped from the stack is: 11

Enter the number to perform following operations
1.Enqueue
2.Dequeue
3.Display
4.Exit
3
        42      23

Enter the number to perform following operations
1.Enqueue
2.Dequeue
3.Display
4.Exit
```

```
File    Line  Message
              === Build file: "no target" in "no project" (compiler: unknown) ===
              === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

Lab Program 7:

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
d) Display the contents of the list

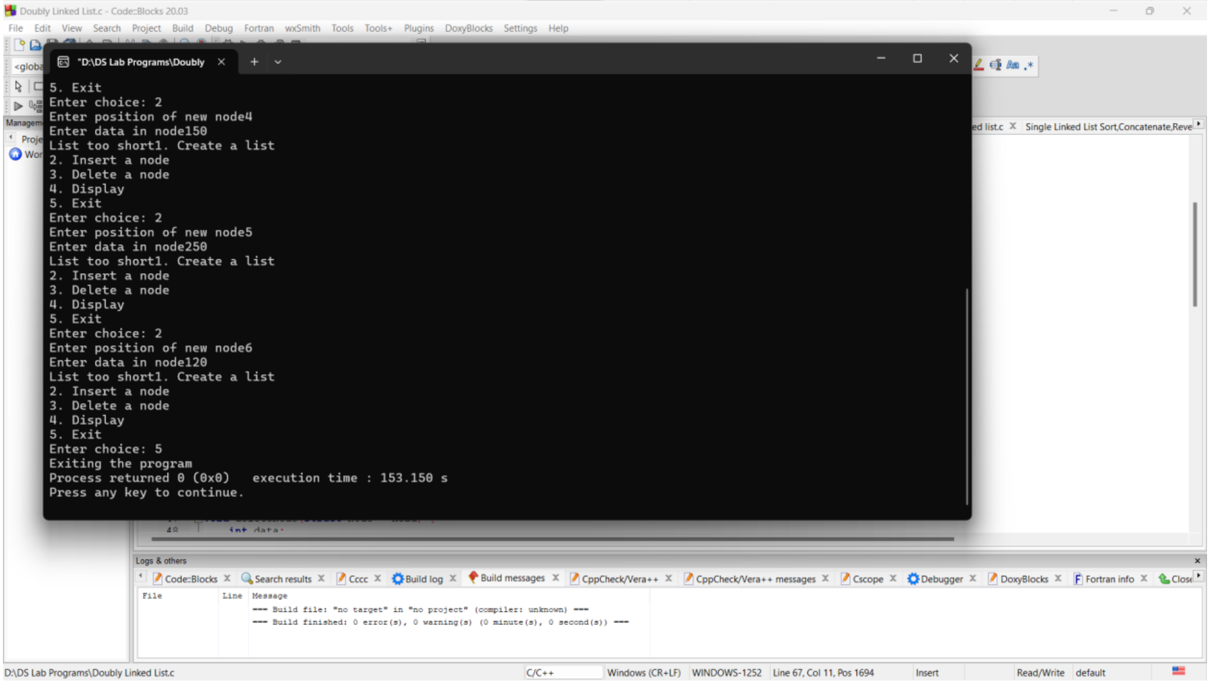
```
#include <stdio.h>
#include<stdlib.h>

struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

struct Node* createNode() {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    int data;
    printf("Enter data in node");
    scanf("%d", &data);
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertNode(struct Node* *head) {
    int pos;
    printf("Enter position of new node");
    scanf("%d", &pos);
    struct Node* newNode = createNode();
    struct Node* temp = (*head);
    while(--pos){
        if(temp->next != NULL)
            temp = temp->next;
        else{
            printf("List too short");
            return;
        }
    }
    if (temp->next == NULL) {
        newNode->next = *head;
        (*head)->prev = newNode;
        (*head) = newNode;
    } else {
        temp->prev->next = newNode;
        newNode->prev = temp->prev;
        temp->prev = newNode;
        newNode->next = temp;
    }
}

void deleteNode(struct Node* *head) {
    int data;
    printf("Enter data in node to be deleted");
    scanf("%d", &data);
    struct Node* current = *head;
    while (current != NULL) {
        if (current->data == data) {
            if (current->prev != NULL) {
                current->prev->next = current->next;
            } else {
                *head = current->next;
            }
            if (current->next != NULL) {
                current->next->prev = current->prev;
            }
            free(current);
            return;
        }
        current = current->next;
    }
    printf("Node with value %d not found", data);
}

void display(struct Node* head) {
    struct Node* current = head;
    printf("Doubly Linked List: ");
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    int choice;
    struct Node* head = NULL;
    while (1) {
        printf("1. Create a list\n");
        printf("2. Insert a node\n");
        printf("3. Delete a node\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                head = createNode();
                break;
            case 2:
                insertNode(&head);
                break;
            case 3:
                deleteNode(&head);
                break;
            case 4:
                display(head);
                break;
            default:
                printf("Exiting the program");
                return 0;
        }
    }
}
```



```
1. Create a list
2. Insert a node
3. Delete a node
4. Display
5. Exit
Enter choice: 1
Enter data in node45
1. Create a list
2. Insert a node
3. Delete a node
4. Display
5. Exit
Enter choice: 2
Enter position of new node2
Enter data in node100
List too short1. Create a list
2. Insert a node
3. Delete a node
4. Display
5. Exit
Enter choice: 2
Enter position of new node3
Enter data in node200
List too short1. Create a list
2. Insert a node
3. Delete a node
4. Display
5. Exit
Enter choice: |
```

```
File    Line    Message
                === Build file: "no target" in "no project" (compiler: unknown) ===
                === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

Doubly Linked List.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\DS Lab Programs\Doubly
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close
D:\DS Lab Programs\Doubly Linked List.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default



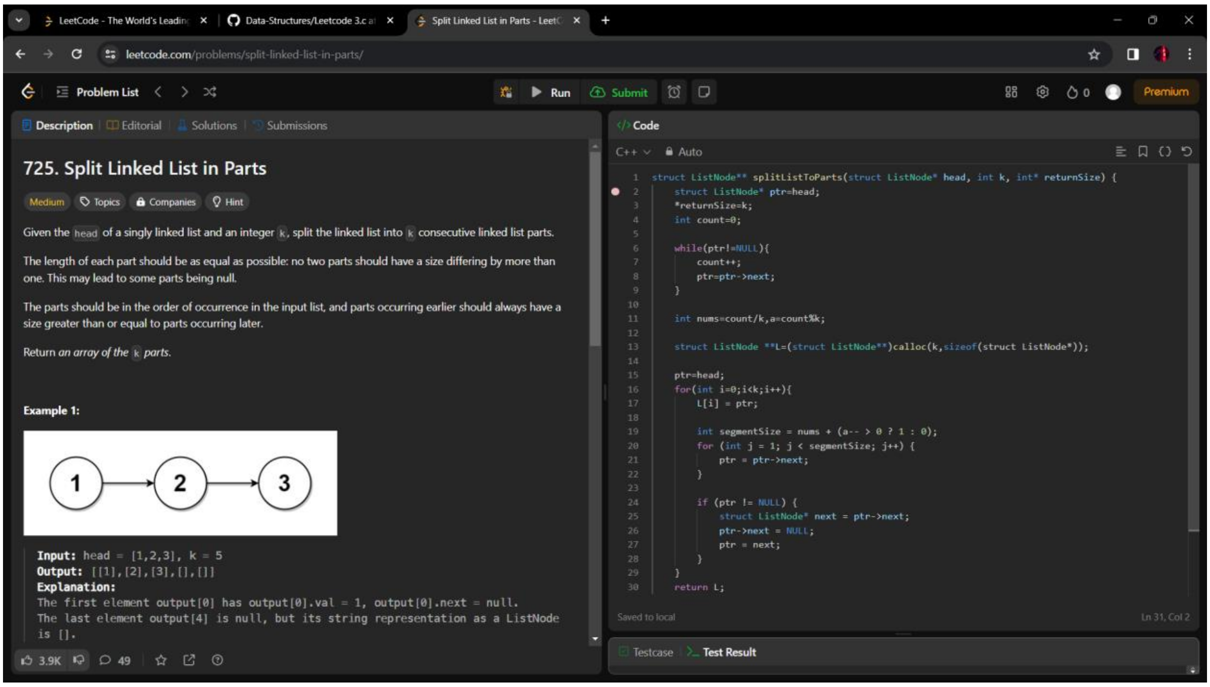
```
5. Exit
Enter choice: 2
Enter position of new node4
Enter data in node150
List too short1. Create a list
2. Insert a node
3. Delete a node
4. Display
5. Exit
Enter choice: 2
Enter position of new node5
Enter data in node250
List too short1. Create a list
2. Insert a node
3. Delete a node
4. Display
5. Exit
Enter choice: 2
Enter position of new node6
Enter data in node120
List too short1. Create a list
2. Insert a node
3. Delete a node
4. Display
5. Exit
Enter choice: 5
Exiting the program
Process returned 0 (0x0)   execution time : 153.150 s
Press any key to continue.
```

```
File    Line    Message
                === Build file: "no target" in "no project" (compiler: unknown) ===
                === Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===
```

Doubly Linked List.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
"D:\DS Lab Programs\Doubly
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks Fortran info Close
D:\DS Lab Programs\Doubly Linked List.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 67, Col 11, Pos 1694 Insert Read/Write default

Leetcode Program: Split Linked List

```
struct ListNode** splitListToParts(struct ListNode* head, int k, int* returnSize) {
    struct ListNode* ptr=head;
    *returnSize=k;
    int count=0;

    while(ptr!=NULL){
        count++;
        ptr=ptr->next;
    }
    int nums=count/k,a=count%k;

    struct ListNode **L=(struct ListNode**)calloc(k,sizeof(struct ListNode*));

    ptr=head;
    for(int i=0;i<k;i++){
        L[i] = ptr;

        int segmentSize = nums + (a-- > 0 ? 1 : 0);
        for (int j = 1; j < segmentSize; j++) {
            ptr = ptr->next;
        }

        if (ptr != NULL) {
            struct ListNode* next = ptr->next;
            ptr->next = NULL;
            ptr = next;
        }
    }
    return L;
}
```



725. Split Linked List in Parts

Given the head of a singly linked list and an integer k, split the linked list into k consecutive linked list parts.

The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Return *an array of the* k *parts.*

Example 1:

```
Input: head = [1,2,3], k = 5
Output: [[1],[2],[3],[],[]]
Explanation:
The first element output[0] has output[0].val = 1, output[0].next = null.
The last element output[4] is null, but its string representation as a ListNode
is [].
```

```
struct ListNode** splitListToParts(struct ListNode* head, int k, int* returnSize) {
    struct ListNode* ptr=head;
    *returnSize=k;
    int count=0;

    while(ptr!=NULL){
        count++;
        ptr=ptr->next;
    }

    int nums=count/k,a=count%k;

    struct ListNode **L=(struct ListNode**)calloc(k,sizeof(struct ListNode*));

    ptr=head;
    for(int i=0;i<k;i++){
        L[i] = ptr;

        int segmentSize = nums + (a-- > 0 ? 1 : 0);
        for (int j = 1; j < segmentSize; j++) {
            ptr = ptr->next;
        }

        if (ptr != NULL) {
            struct ListNode* next = ptr->next;
            ptr->next = NULL;
            ptr = next;
        }
    }
    return L;
```

Lab Program 8:

Write a program

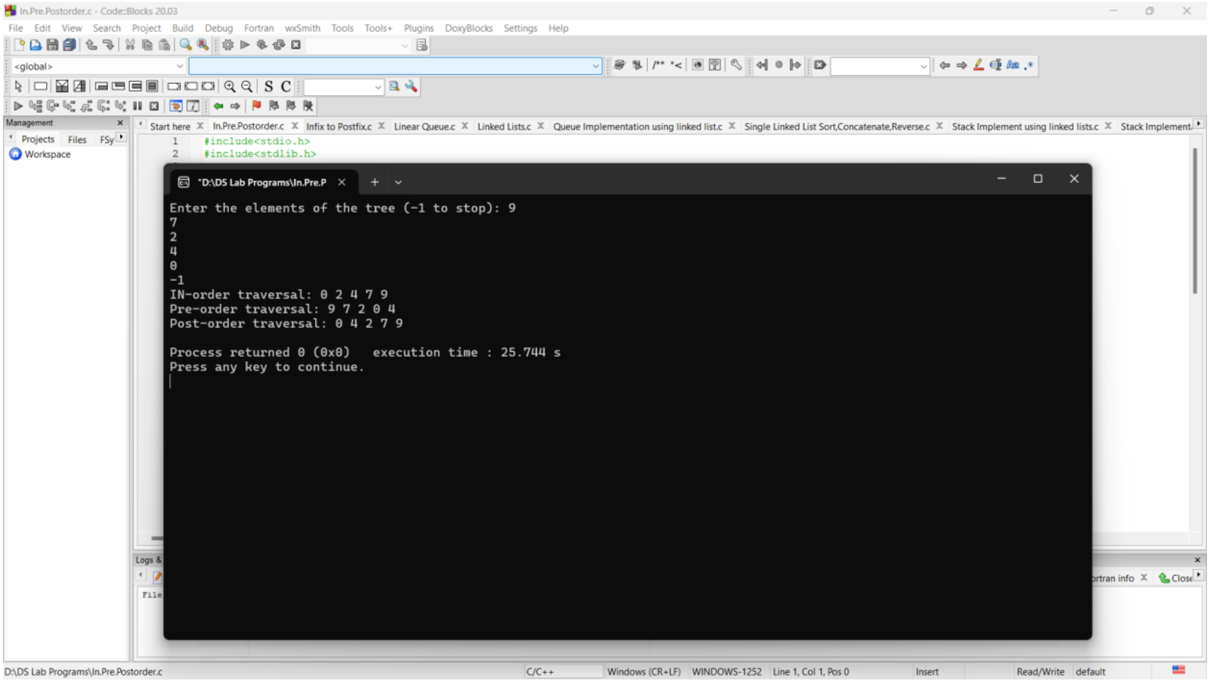
a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.

```
#include<stdio.h>
#include<stdlib.h>
struct node{
int data;
struct node*left;
struct node*right;
};
struct node*newnode(int data){
struct node*node=(struct node*)malloc(sizeof(struct node));
node->data=data;
node->left=NULL;
node->right=NULL;
return node;
}
struct node* insert(struct node* root,int data){
if (root=NULL)
    {
        return newnode(data);
    }
    else{
        if(data<root->data)
        {
            root->right=insert(root->right,data);
        }
    }return root;
}
void inorder(struct node *root)
{
    if (root!=NULL)
    {
      inorder(root->left);
    printf("%d",root->data);
    inorder(root->right);
    }
}
void preorder(struct node* root)
{
    if (root!=NULL)
    {
        printf("%d",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
```

```
void postorder(struct node* root)
{
   if(root!=NULL)
   {
      postorder(root->left);
      postorder(root->right);
      printf("%d",root->data);
   }
}
void display(struct node* root)
{
   if (root!=NULL)
   {
      printf("IN-order traversal:");
      inorder(root);
      printf("\n");
      printf("Pre-order traversal:");
      preorder(root);
      printf("\n");
      printf("Post-order traversal:");
      postorder(root);
      printf("\n");
   }
}
int main()
{
   struct node*root=NULL;
   int data;
   printf("Enter the elements of the tree(-1 to stop):");
   while(scanf("%d",&data)&&data!=-1)
   {
      root=insert(root,data);
   }display(root);
   return 0;
}
```

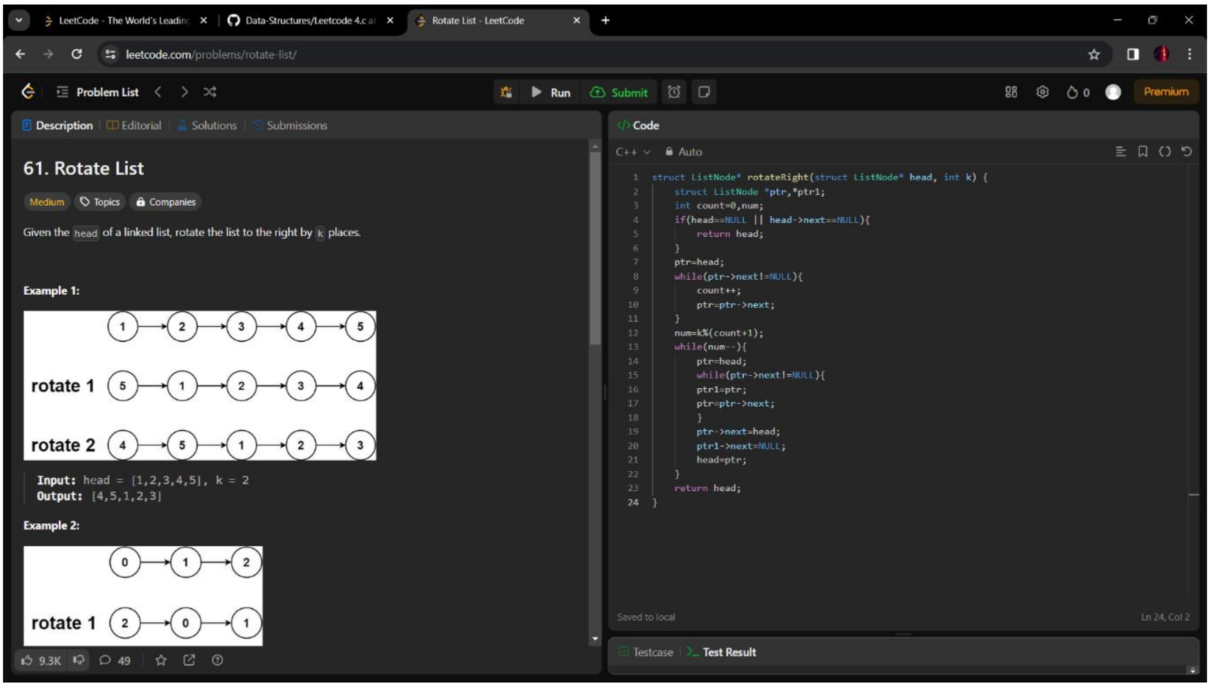

In.Pre.Postorder.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
In.Pre.Postorder.c
Infix to Postfix.c
Linear Queue.c
Linked Lists.c
Queue Implementation using linked list.c
Single Linked List Sort,Concatenate,Reverse.c
Stack Implement using linked lists.c
"D:\DS Lab Programs\In.Pre.P
D:\DS Lab Programs\In.Pre.Postorder.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

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

```
Enter the elements of the tree (-1 to stop): 9
7
2
4
0
-1
IN-order traversal: 0 2 4 7 9
Pre-order traversal: 9 7 2 0 4
Post-order traversal: 0 4 2 7 9

Process returned 0 (0x0)   execution time : 25.744 s
Press any key to continue.
```

Leetcode Program: Rotate List

```c
struct ListNode* rotateRight(struct ListNode* head, int k) {
    struct ListNode *ptr,*ptr1;
    int count=0,num;
    if(head==NULL || head->next==NULL){
        return head;
    }
    ptr=head;
    while(ptr->next!=NULL){
        count++;
        ptr=ptr->next;
    }
    num=k%(count+1);
    while(num--){
        ptr=head;
        while(ptr->next!=NULL){
        ptr1=ptr;
        ptr=ptr->next;
        }
        ptr->next=head;
        ptr1->next=NULL;
        head=ptr;
    }
    return head;
}
```



61. Rotate List

Given the head of a linked list, rotate the list to the right by k places.

Example 1:

```
Input: head = [1,2,3,4,5], k = 2
Output: [4,5,1,2,3]
```

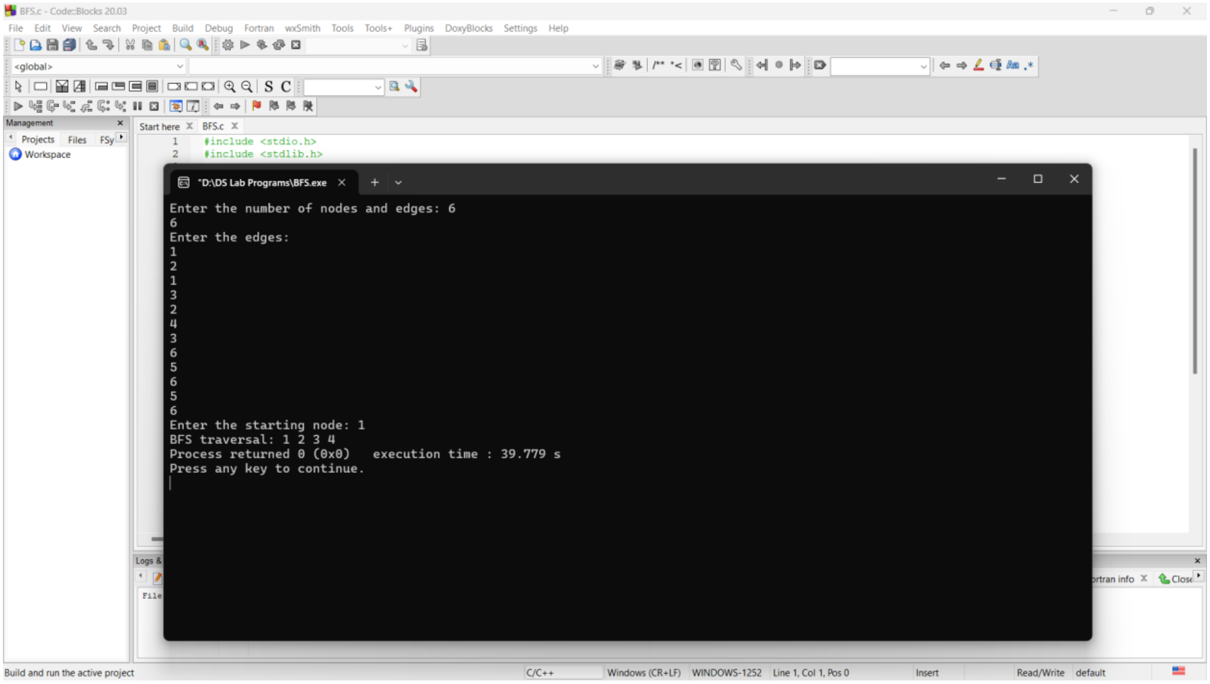
Example 2:

```
struct ListNode* rotateRight(struct ListNode* head, int k) {
    struct ListNode *ptr,*ptr1;
    int count=0,num;
    if(head==NULL || head->next==NULL){
        return head;
    }
    ptr=head;
    while(ptr->next!=NULL){
        count++;
        ptr=ptr->next;
    }
    num=k%(count+1);
    while(num--){
        ptr=head;
        while(ptr->next!=NULL){
        ptr1=ptr;
        ptr=ptr->next;
        }
        ptr->next=head;
        ptr1->next=NULL;
        head=ptr;
    }
    return head;
}
```

Lab Program 9:
9a) Write a program to traverse a graph using BFS method.
9b) Write a program to check whether given graph is connected or not using DFS method.

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

#define MAX_NODES 100
#define MAX_EDGES 100

int graph[MAX_NODES][MAX_NODES];
int visited[MAX_NODES];
int queue[MAX_NODES];
int front = -1, rear = -1;

void BFS(int start, int n) {
   visited[start] = 1;
   queue[++rear] = start;

   while(front != rear) {
      int current = queue[++front];
      printf("%d ", current);

      for(int i = 0; i < n; i++) {
         if(graph[current][i] == 1 && !visited[i]) {
            visited[i] = 1;
```

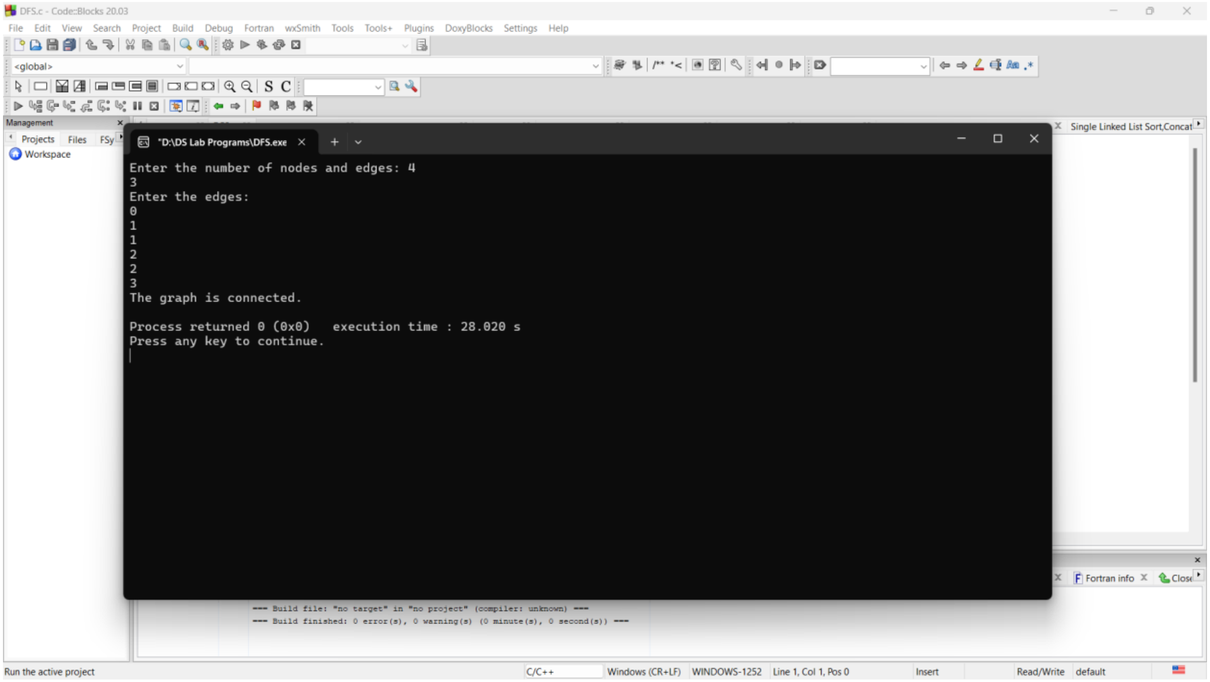
```
                queue[++rear] = i;
            }
        }
    }
}

int main() {
    int n, m;
    printf("Enter the number of nodes and edges: ");
    scanf("%d %d", &n, &m);

    printf("Enter the edges:\n");
    for(int i = 0; i < m; i++) {
        int a, b;
        scanf("%d %d", &a, &b);
        graph[a][b] = 1;
        graph[b][a] = 1;
    }

    int start;
    printf("Enter the starting node: ");
    scanf("%d", &start);

    printf("BFS traversal: ");
    BFS(start, n);

    return 0;
}
```


BFS.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSy
Workspace
Start here
BFS.c
"D:\DS Lab Programs\BFS.exe
Logs &
Build and run the active project
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

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

```
Enter the number of nodes and edges: 6
6
Enter the edges:
1
2
1
3
2
4
3
6
5
6
5
6
Enter the starting node: 1
BFS traversal: 1 2 3 4
Process returned 0 (0x0)   execution time : 39.779 s
Press any key to continue.
```

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

int arr[20][20];
int visited[20];

void dfs(int start, int n) {
    visited[start] = 1;
    for(int i = 0; i < n; i++) {
        if(arr[start][i] == 1 && !visited[i]) {
            dfs(i, n);
        }
    }
}

int isConnected(int n) {
    dfs(0, n);

    for(int i = 0; i < n; i++) {
        if(!visited[i]) {
            return 0;
        }
    }

    return 1;
}

int main() {
    int n, m;
    printf("Enter the number of nodes and edges: ");
    scanf("%d %d", &n, &m);

    printf("Enter the edges:\n");
    for(int i = 0; i < m; i++) {
        int a, b;
        scanf("%d %d", &a, &b);
        arr[a][b] = 1;
        arr[b][a] = 1;
    }

    if(isConnected(n)) {
        printf("The graph is connected.\n");
    } else {
        printf("The graph is not connected.\n");
    }

    return 0;
}
```



DFS.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FSy
Workspace
"D:\DS Lab Programs\DFS.exe
Single Linked List Sort,Concat
Fortran info
Close
Run the active project
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default

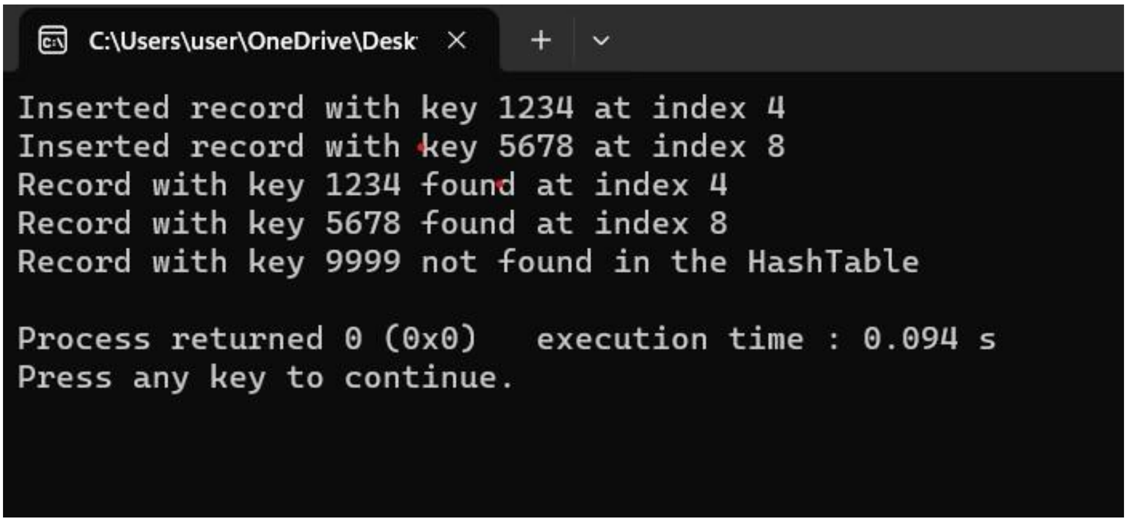
```
Enter the number of nodes and edges: 4
3
Enter the edges:
0
1
1
2
2
3
The graph is connected.

Process returned 0 (0x0)   execution time : 28.020 s
Press any key to continue.
```

```
--- Build file: "no target" in "no project" (compiler: unknown) ---
--- Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ---
```

Lab Program 10: Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are integers. Design and develop a Program in C that uses Hash function H: K -> L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

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

#define TABLE_SIZE 10

struct Employee {
    int key;
};
```

```
struct HashTable {
   struct Employee* table[TABLE_SIZE];
};

void initializeHashTable(struct HashTable* ht) {
   for (int i = 0; i < TABLE_SIZE; i++) {
      ht->table[i] = NULL;
   }
}

int hashFunction(int key) {
   return key % TABLE_SIZE;
}


void insert(struct HashTable* ht, int key, struct Employee* emp) {
   int hkey = hashFunction(key);
   int index = hkey;
   int i = 0;

   while (ht->table[index] != NULL) {
      i++;
      index = (hkey + i) % TABLE_SIZE;
   }

   ht->table[index] = emp;
}

struct Employee* search(struct HashTable* ht, int key) {
   int hkey = hashFunction(key);
   int index = hkey;
   int i = 0;

   while (ht->table[index] != NULL) {
      if (ht->table[index]->key == key) {
         return ht->table[index];
      }

      i++;
      index = (hkey + i) % TABLE_SIZE;
   }

   return NULL;
}

void displayHashTable(struct HashTable* ht) {
   printf("\nHash Table:\n");
   for (int i = 0; i < TABLE_SIZE; i++) {
      if (ht->table[i] != NULL) {
         printf("Index %d: Key %d\n", i, ht->table[i]->key);
      }
   }
}

int main() {
   struct HashTable ht;
   initializeHashTable(&ht);

   struct Employee emp1 = {101};
   struct Employee emp2 = {201};
   struct Employee emp3 = {301};

   insert(&ht, emp1.key, &emp1);
   insert(&ht, emp2.key, &emp2);
   insert(&ht, emp3.key, &emp3);

   displayHashTable(&ht);

   int searchKey = 201;
   struct Employee* result = search(&ht, searchKey);

   if (result != NULL) {
      printf("\nEmployee with key %d found!\n", searchKey);
   } else {
      printf("\nEmployee with key %d not found!\n", searchKey);
   }

   return 0;
}
```



```
Inserted record with key 1234 at index 4
Inserted record with key 5678 at index 8
Record with key 1234 found at index 4
Record with key 5678 found at index 8
Record with key 9999 not found in the HashTable

Process returned 0 (0x0)   execution time : 0.094 s
Press any key to continue.
```