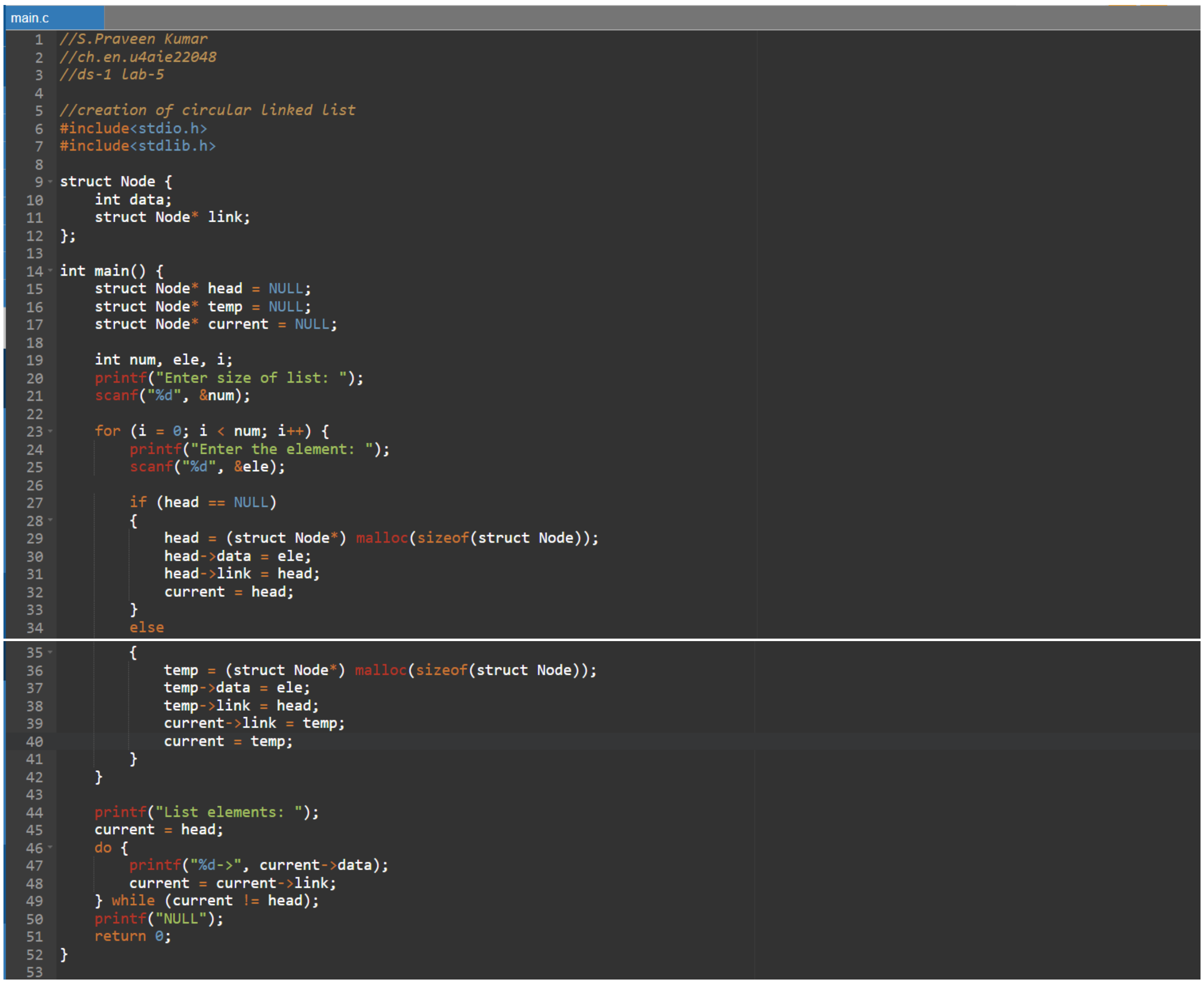
# DATA STRUCTURE – 1

# LAB-5

S.Praveen kumar
ch.en.u4aie22048

## Creation of circular linked list:

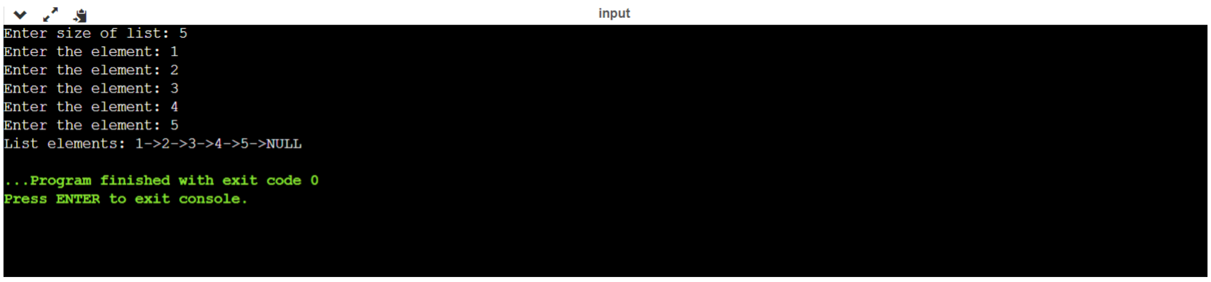
### Program:



main.c

```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-5

//creation of circular linked list
#include<stdio.h>
#include<stdlib.h>

struct Node {
    int data;
    struct Node* link;
};

int main() {
    struct Node* head = NULL;
    struct Node* temp = NULL;
    struct Node* current = NULL;

    int num, ele, i;
    printf("Enter size of list: ");
    scanf("%d", &num);

    for (i = 0; i < num; i++) {
        printf("Enter the element: ");
        scanf("%d", &ele);

        if (head == NULL)
        {
            head = (struct Node*) malloc(sizeof(struct Node));
            head->data = ele;
            head->link = head;
            current = head;
        }
        else
        {
            temp = (struct Node*) malloc(sizeof(struct Node));
            temp->data = ele;
            temp->link = head;
            current->link = temp;
            current = temp;
        }
    }

    printf("List elements: ");
    current = head;
    do {
        printf("%d->", current->data);
        current = current->link;
    } while (current != head);
    printf("NULL");
    return 0;
}
```

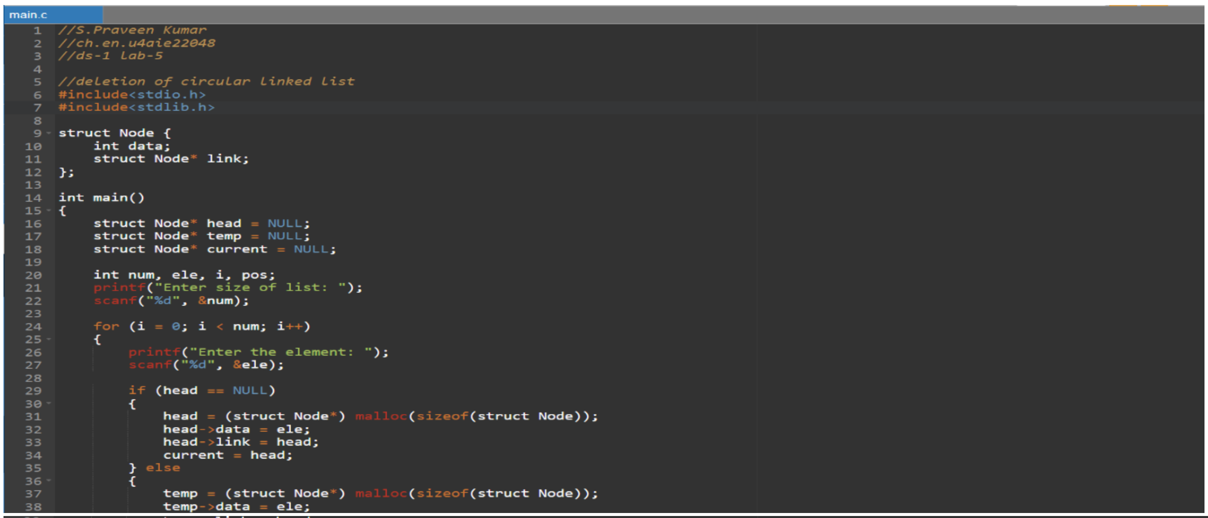
### Output:



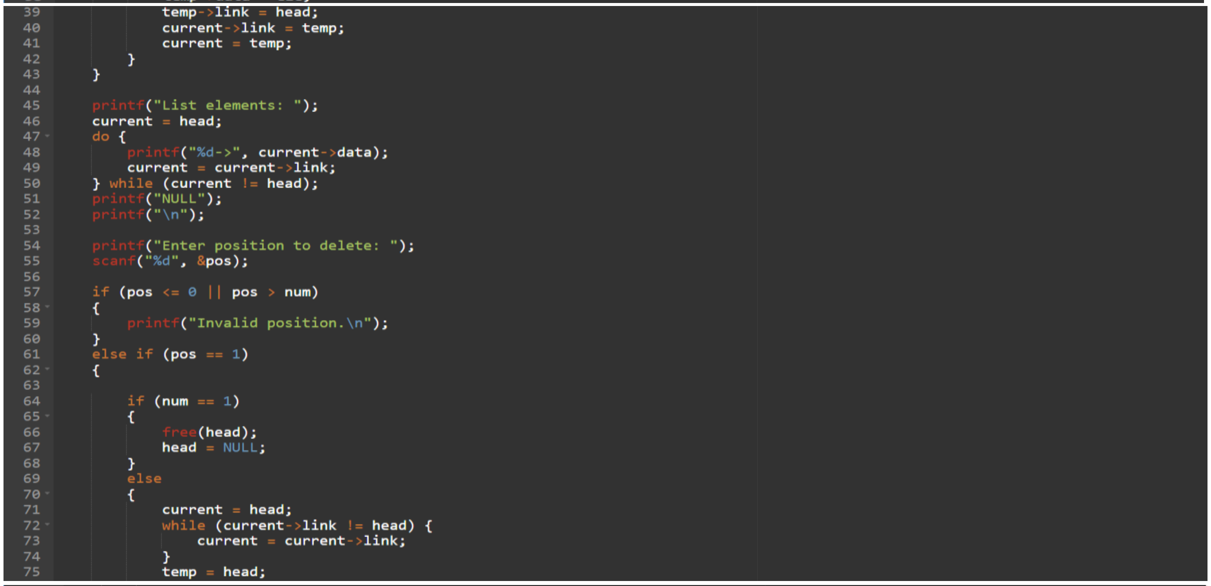
```
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 4
Enter the element: 5
List elements: 1->2->3->4->5->NULL

...Program finished with exit code 0
Press ENTER to exit console.
```

# Deletion of circular linked list:

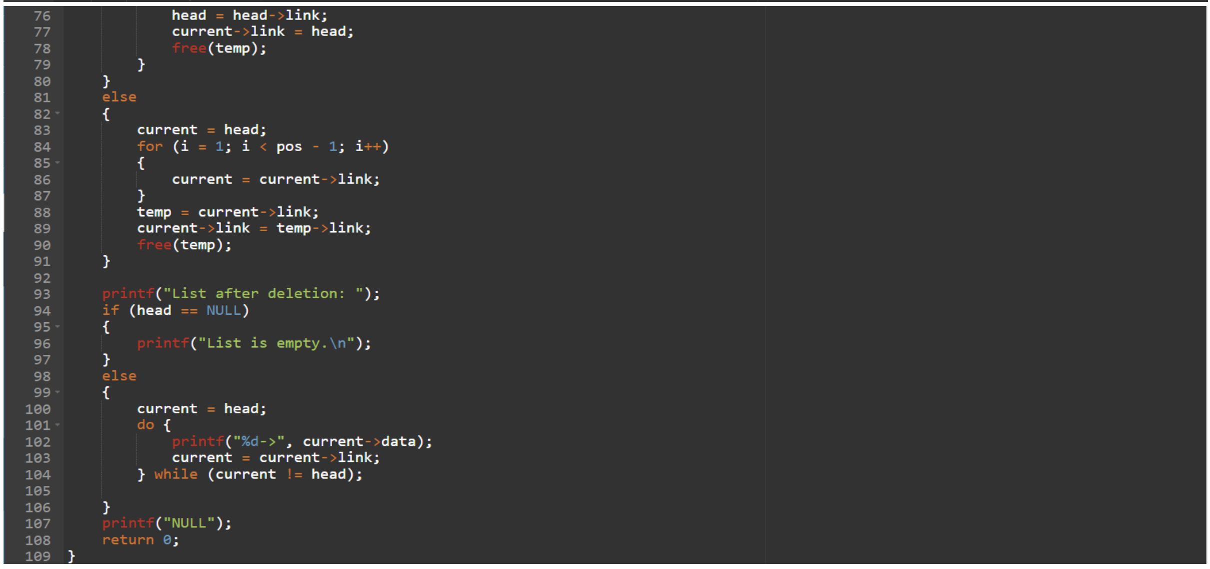
## Program:



```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 Lab-5

//deletion of circular linked list
#include<stdio.h>
#include<stdlib.h>

struct Node {
    int data;
    struct Node* link;
};

int main()
{
    struct Node* head = NULL;
    struct Node* temp = NULL;
    struct Node* current = NULL;

    int num, ele, i, pos;
    printf("Enter size of list: ");
    scanf("%d", &num);

    for (i = 0; i < num; i++)
    {
        printf("Enter the element: ");
        scanf("%d", &ele);

        if (head == NULL)
        {
            head = (struct Node*) malloc(sizeof(struct Node));
            head->data = ele;
            head->link = head;
            current = head;
        } else
        {
            temp = (struct Node*) malloc(sizeof(struct Node));
            temp->data = ele;
```

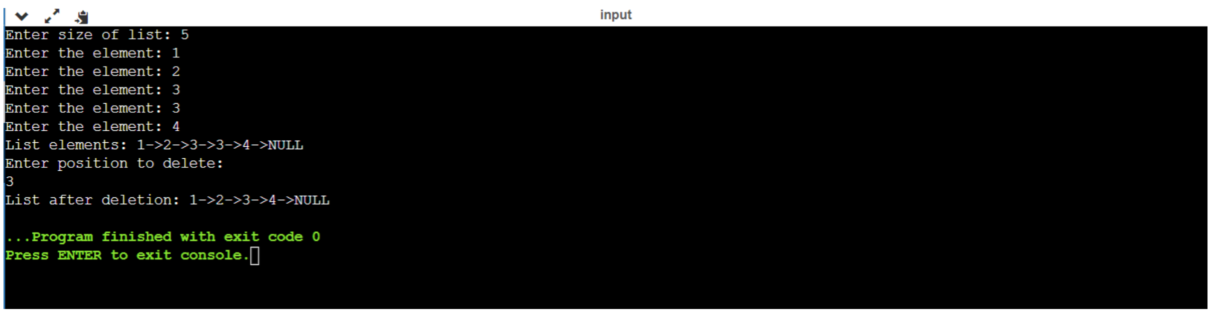


```c
            temp->link = head;
            current->link = temp;
            current = temp;
        }
    }

    printf("List elements: ");
    current = head;
    do {
        printf("%d->", current->data);
        current = current->link;
    } while (current != head);
    printf("NULL");
    printf("\n");

    printf("Enter position to delete: ");
    scanf("%d", &pos);

    if (pos <= 0 || pos > num)
    {
        printf("Invalid position.\n");
    }
    else if (pos == 1)
    {

        if (num == 1)
        {
            free(head);
            head = NULL;
        }
        else
        {
            current = head;
            while (current->link != head) {
                current = current->link;
            }
            temp = head;
```



```c
            head = head->link;
            current->link = head;
            free(temp);
        }
    }
    else
    {
        current = head;
        for (i = 1; i < pos - 1; i++)
        {
            current = current->link;
        }
        temp = current->link;
        current->link = temp->link;
        free(temp);
    }

    printf("List after deletion: ");
    if (head == NULL)
    {
        printf("List is empty.\n");
    }
    else
    {
        current = head;
        do {
            printf("%d->", current->data);
            current = current->link;
        } while (current != head);

    }
    printf("NULL");
    return 0;
}
```

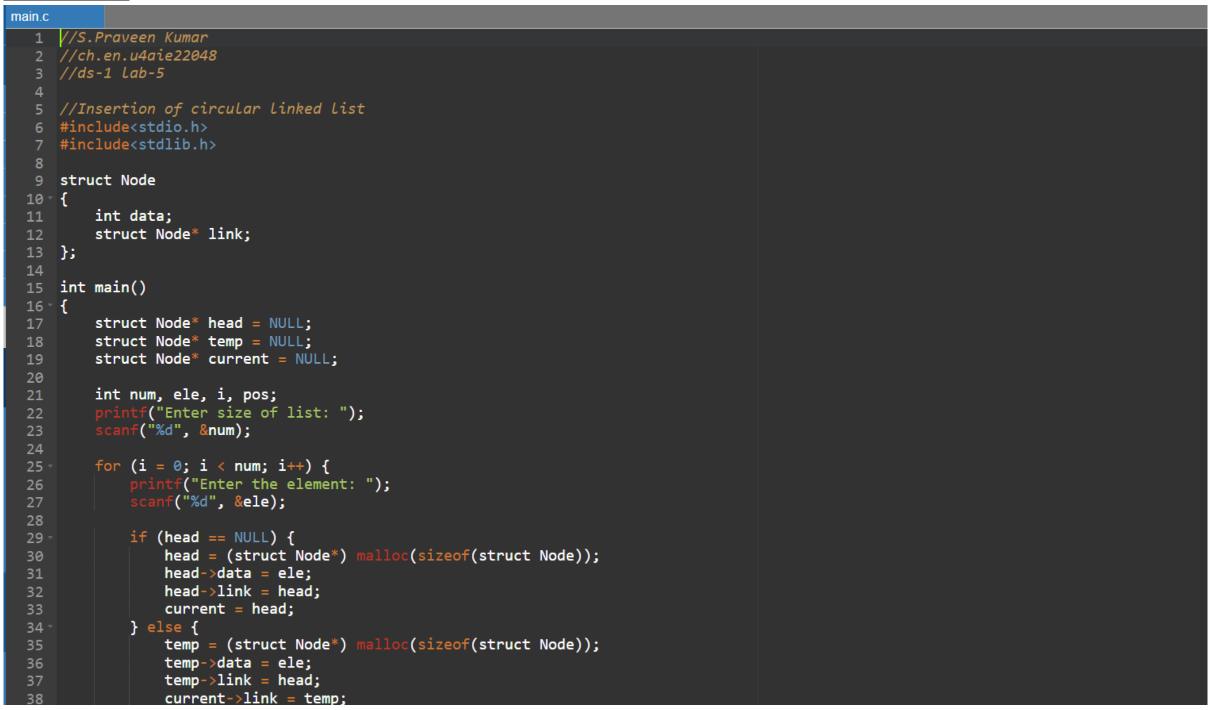
## Output:



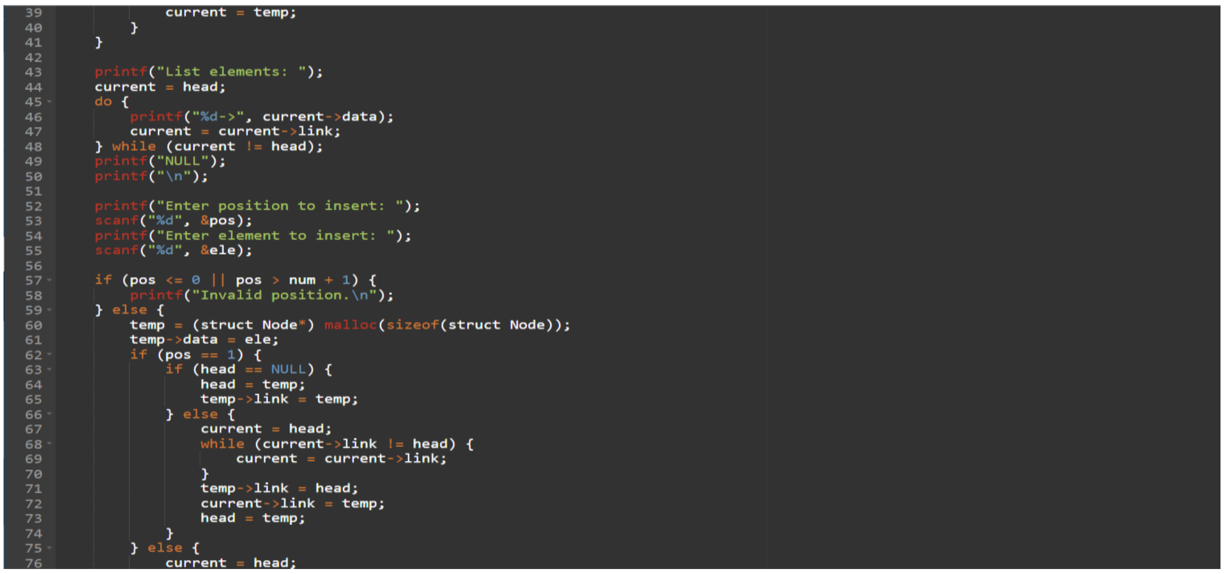
```
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 3
Enter the element: 4
List elements: 1->2->3->3->4->NULL
Enter position to delete:
3
List after deletion: 1->2->3->4->NULL

...Program finished with exit code 0
Press ENTER to exit console.
```

# Insertion of circular linked list:

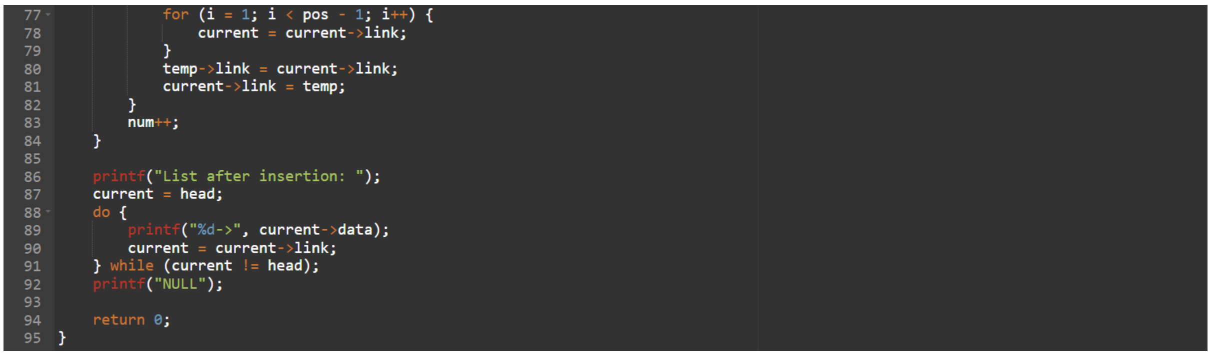
## Program:



```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-5

//Insertion of circular linked list
#include<stdio.h>
#include<stdlib.h>

struct Node
{
    int data;
    struct Node* link;
};

int main()
{
    struct Node* head = NULL;
    struct Node* temp = NULL;
    struct Node* current = NULL;

    int num, ele, i, pos;
    printf("Enter size of list: ");
    scanf("%d", &num);

    for (i = 0; i < num; i++) {
        printf("Enter the element: ");
        scanf("%d", &ele);

        if (head == NULL) {
            head = (struct Node*) malloc(sizeof(struct Node));
            head->data = ele;
            head->link = head;
            current = head;
        } else {
            temp = (struct Node*) malloc(sizeof(struct Node));
            temp->data = ele;
            temp->link = head;
            current->link = temp;
```

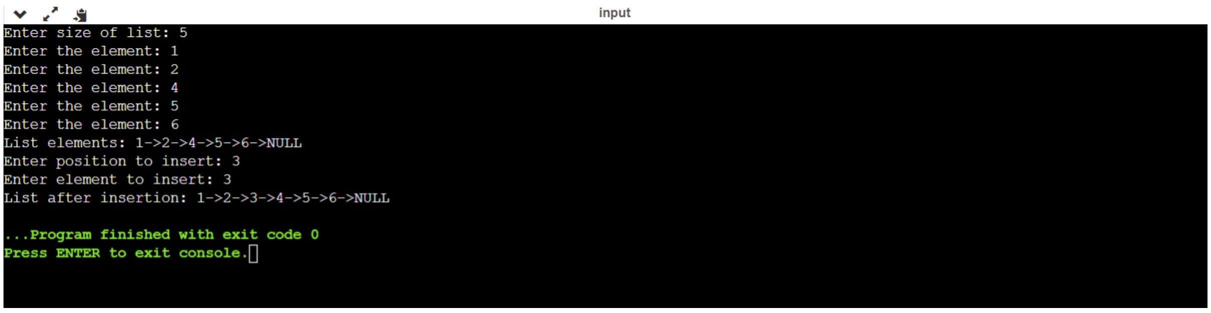


```c
            current = temp;
        }
    }

    printf("List elements: ");
    current = head;
    do {
        printf("%d->", current->data);
        current = current->link;
    } while (current != head);
    printf("NULL");
    printf("\n");

    printf("Enter position to insert: ");
    scanf("%d", &pos);
    printf("Enter element to insert: ");
    scanf("%d", &ele);

    if (pos <= 0 || pos > num + 1) {
        printf("Invalid position.\n");
    } else {
        temp = (struct Node*) malloc(sizeof(struct Node));
        temp->data = ele;
        if (pos == 1) {
            if (head == NULL) {
                head = temp;
                temp->link = temp;
            } else {
                current = head;
                while (current->link != head) {
                    current = current->link;
                }
                temp->link = head;
                current->link = temp;
                head = temp;
            }
        } else {
            current = head;
```



```c
            for (i = 1; i < pos - 1; i++) {
                current = current->link;
            }
            temp->link = current->link;
            current->link = temp;
        }
        num++;
    }

    printf("List after insertion: ");
    current = head;
    do {
        printf("%d->", current->data);
        current = current->link;
    } while (current != head);
    printf("NULL");

    return 0;
}
```

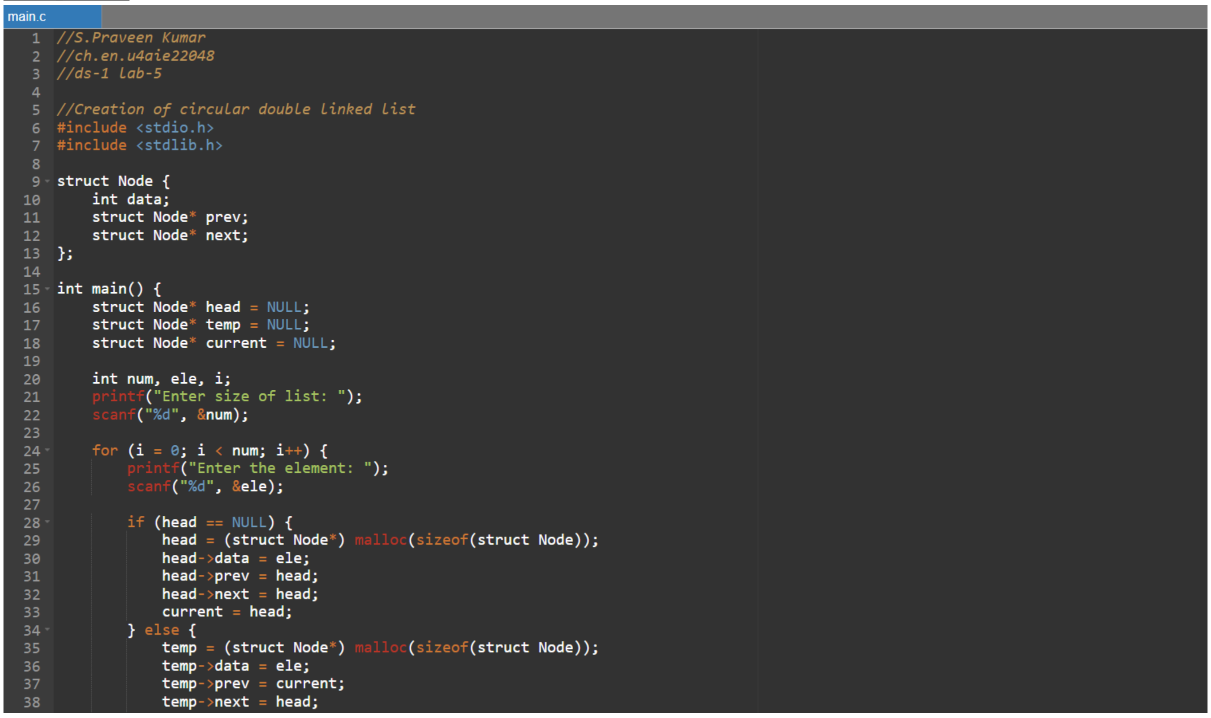
## Output:



```
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 4
Enter the element: 5
Enter the element: 6
List elements: 1->2->4->5->6->NULL
Enter position to insert: 3
Enter element to insert: 3
List after insertion: 1->2->3->4->5->6->NULL

...Program finished with exit code 0
Press ENTER to exit console.
```

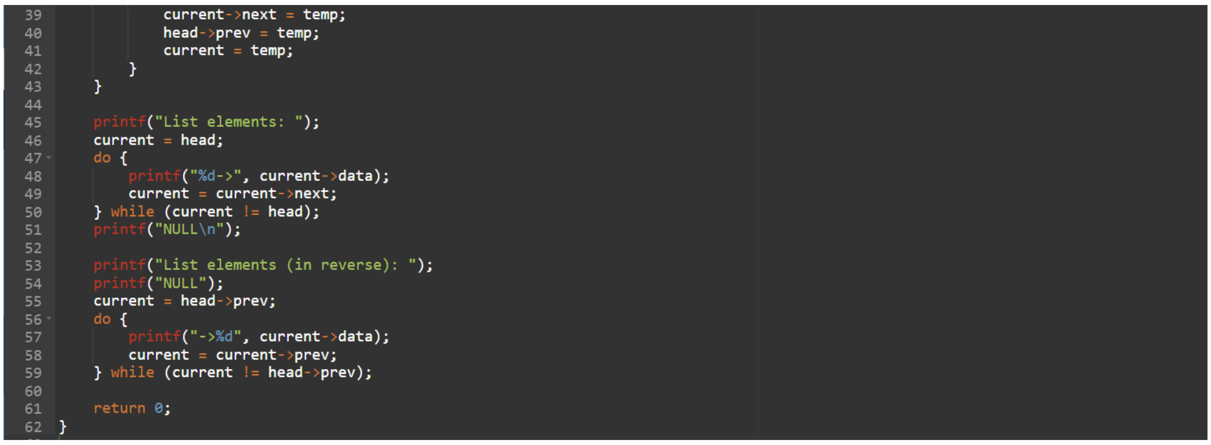
## Creation of circular double linked list:

## Program:



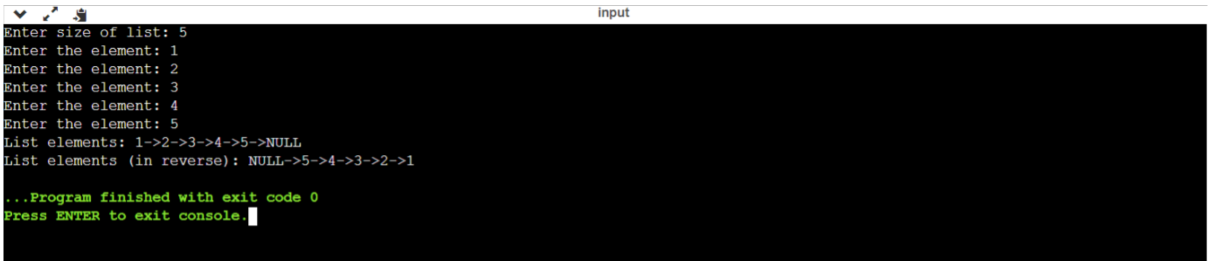
main.c

```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-5

//Creation of circular double linked list
#include <stdio.h>
#include <stdlib.h>

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

int main() {
    struct Node* head = NULL;
    struct Node* temp = NULL;
    struct Node* current = NULL;

    int num, ele, i;
    printf("Enter size of list: ");
    scanf("%d", &num);

    for (i = 0; i < num; i++) {
        printf("Enter the element: ");
        scanf("%d", &ele);

        if (head == NULL) {
            head = (struct Node*) malloc(sizeof(struct Node));
            head->data = ele;
            head->prev = head;
            head->next = head;
            current = head;
        } else {
            temp = (struct Node*) malloc(sizeof(struct Node));
            temp->data = ele;
            temp->prev = current;
            temp->next = head;
```



```c
            current->next = temp;
            head->prev = temp;
            current = temp;
        }
    }

    printf("List elements: ");
    current = head;
    do {
        printf("%d->", current->data);
        current = current->next;
    } while (current != head);
    printf("NULL\n");

    printf("List elements (in reverse): ");
    printf("NULL");
    current = head->prev;
    do {
        printf("->%d", current->data);
        current = current->prev;
    } while (current != head->prev);

    return 0;
}
```

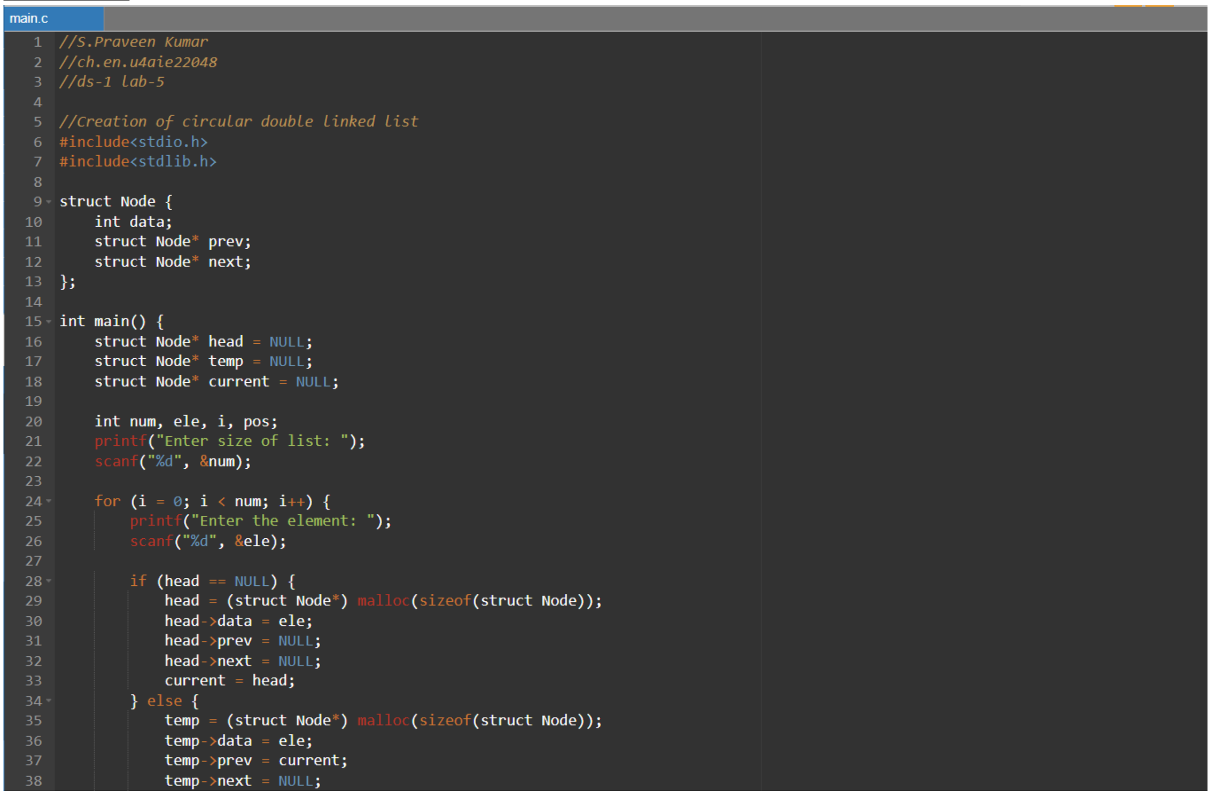
## Output:



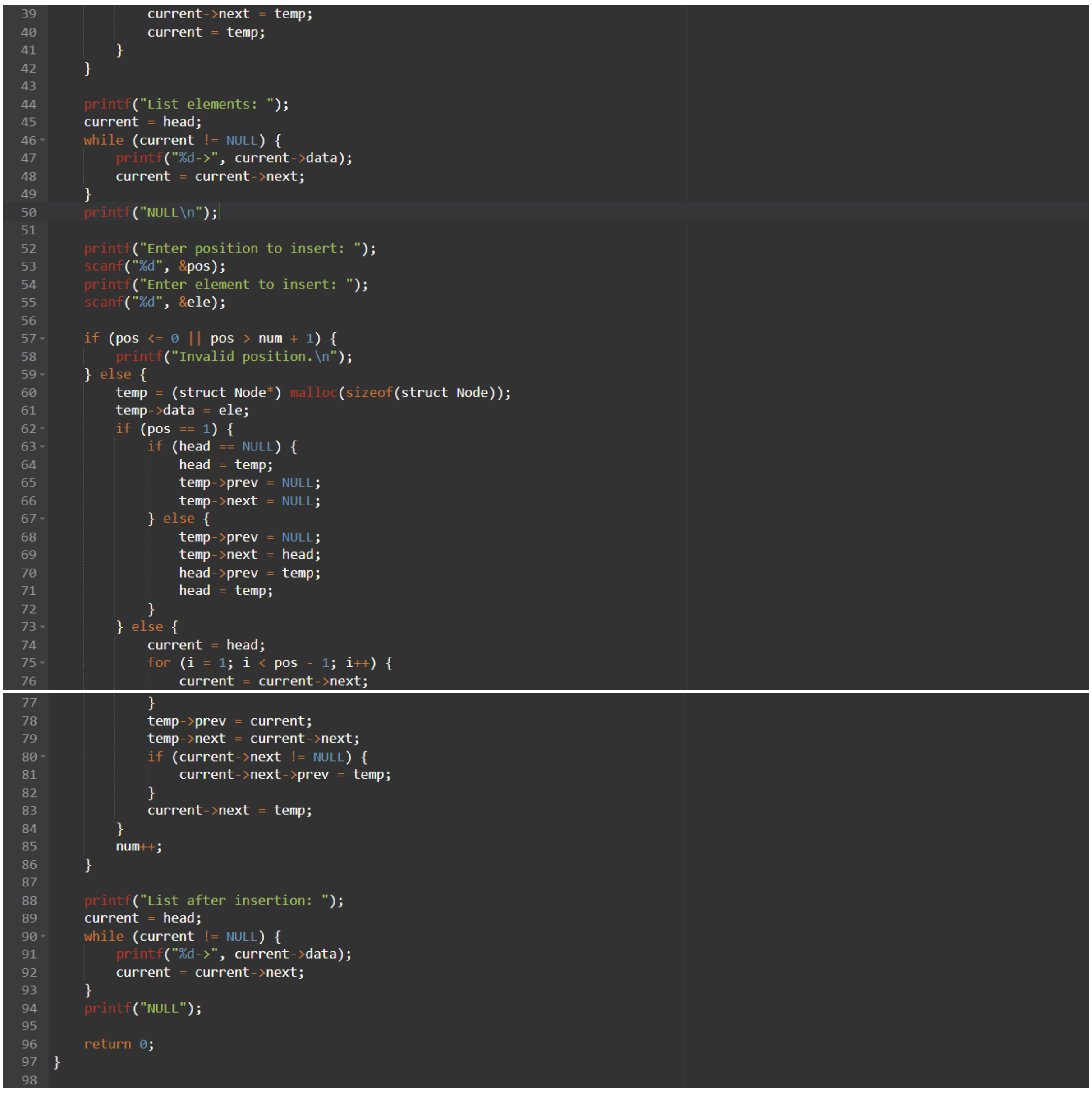
```
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 4
Enter the element: 5
List elements: 1->2->3->4->5->NULL
List elements (in reverse): NULL->5->4->3->2->1

...Program finished with exit code 0
Press ENTER to exit console.
```

## Insertion of circular double linked list:

## Program:

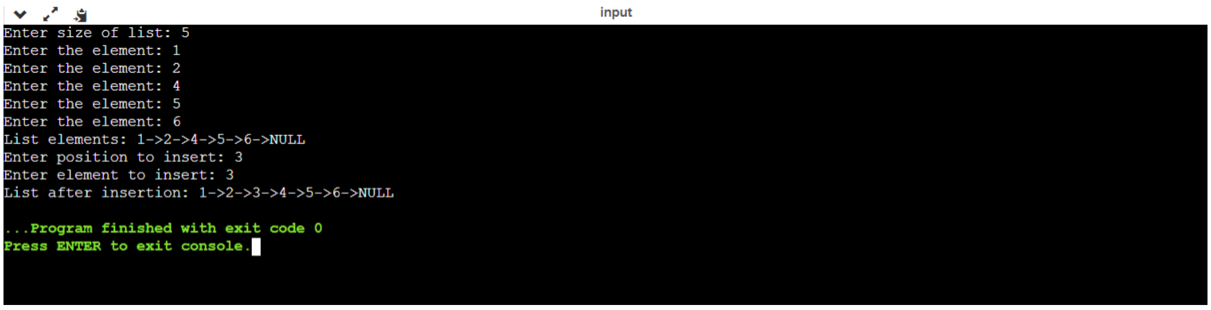


```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-5

//Creation of circular double linked list
#include<stdio.h>
#include<stdlib.h>

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

int main() {
    struct Node* head = NULL;
    struct Node* temp = NULL;
    struct Node* current = NULL;

    int num, ele, i, pos;
    printf("Enter size of list: ");
    scanf("%d", &num);

    for (i = 0; i < num; i++) {
        printf("Enter the element: ");
        scanf("%d", &ele);

        if (head == NULL) {
            head = (struct Node*) malloc(sizeof(struct Node));
            head->data = ele;
            head->prev = NULL;
            head->next = NULL;
            current = head;
        } else {
            temp = (struct Node*) malloc(sizeof(struct Node));
            temp->data = ele;
            temp->prev = current;
            temp->next = NULL;
```



```c
            current->next = temp;
            current = temp;
        }
    }

    printf("List elements: ");
    current = head;
    while (current != NULL) {
        printf("%d->", current->data);
        current = current->next;
    }
    printf("NULL\n");

    printf("Enter position to insert: ");
    scanf("%d", &pos);
    printf("Enter element to insert: ");
    scanf("%d", &ele);

    if (pos <= 0 || pos > num + 1) {
        printf("Invalid position.\n");
    } else {
        temp = (struct Node*) malloc(sizeof(struct Node));
        temp->data = ele;
        if (pos == 1) {
            if (head == NULL) {
                head = temp;
                temp->prev = NULL;
                temp->next = NULL;
            } else {
                temp->prev = NULL;
                temp->next = head;
                head->prev = temp;
                head = temp;
            }
        } else {
            current = head;
            for (i = 1; i < pos - 1; i++) {
                current = current->next;
            }
            temp->prev = current;
            temp->next = current->next;
            if (current->next != NULL) {
                current->next->prev = temp;
            }
            current->next = temp;
        }
        num++;
    }

    printf("List after insertion: ");
    current = head;
    while (current != NULL) {
        printf("%d->", current->data);
        current = current->next;
    }
    printf("NULL");

    return 0;
}
```

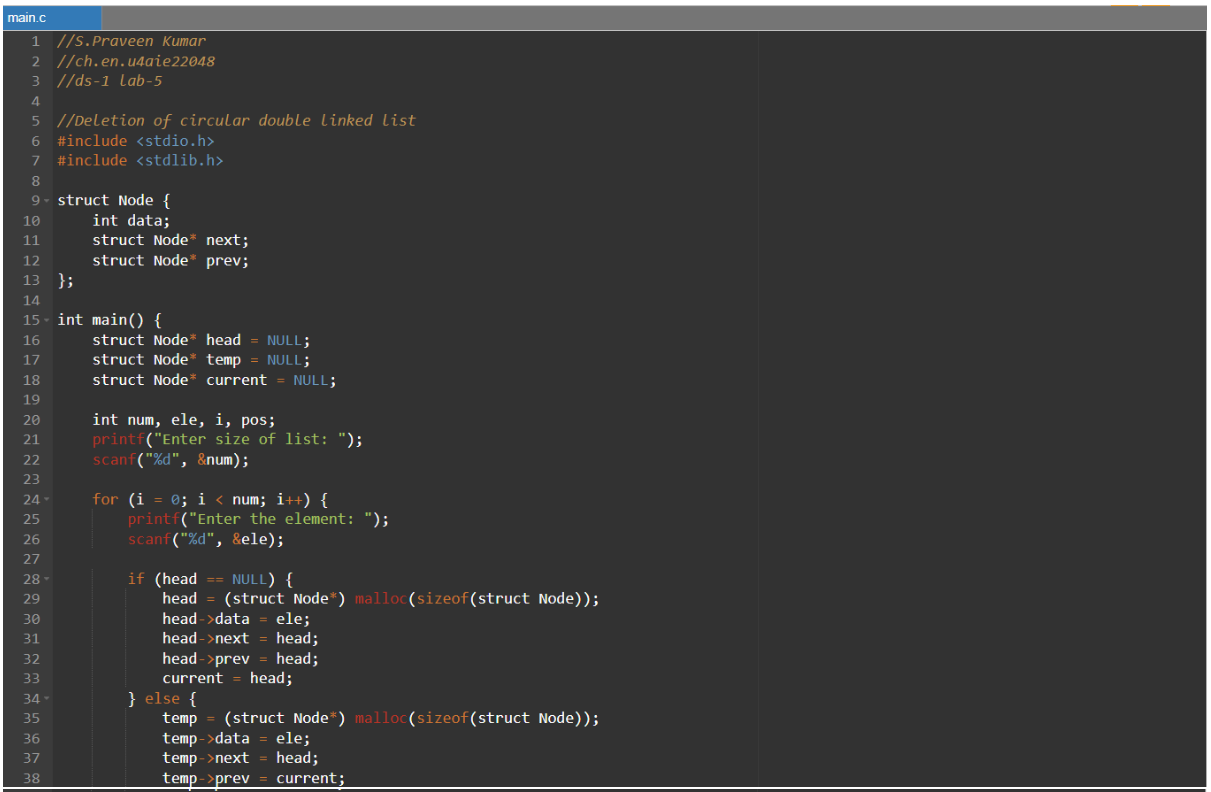
## Output:



```
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 4
Enter the element: 5
Enter the element: 6
List elements: 1->2->4->5->6->NULL
Enter position to insert: 3
Enter element to insert: 3
List after insertion: 1->2->3->4->5->6->NULL

...Program finished with exit code 0
Press ENTER to exit console.
```

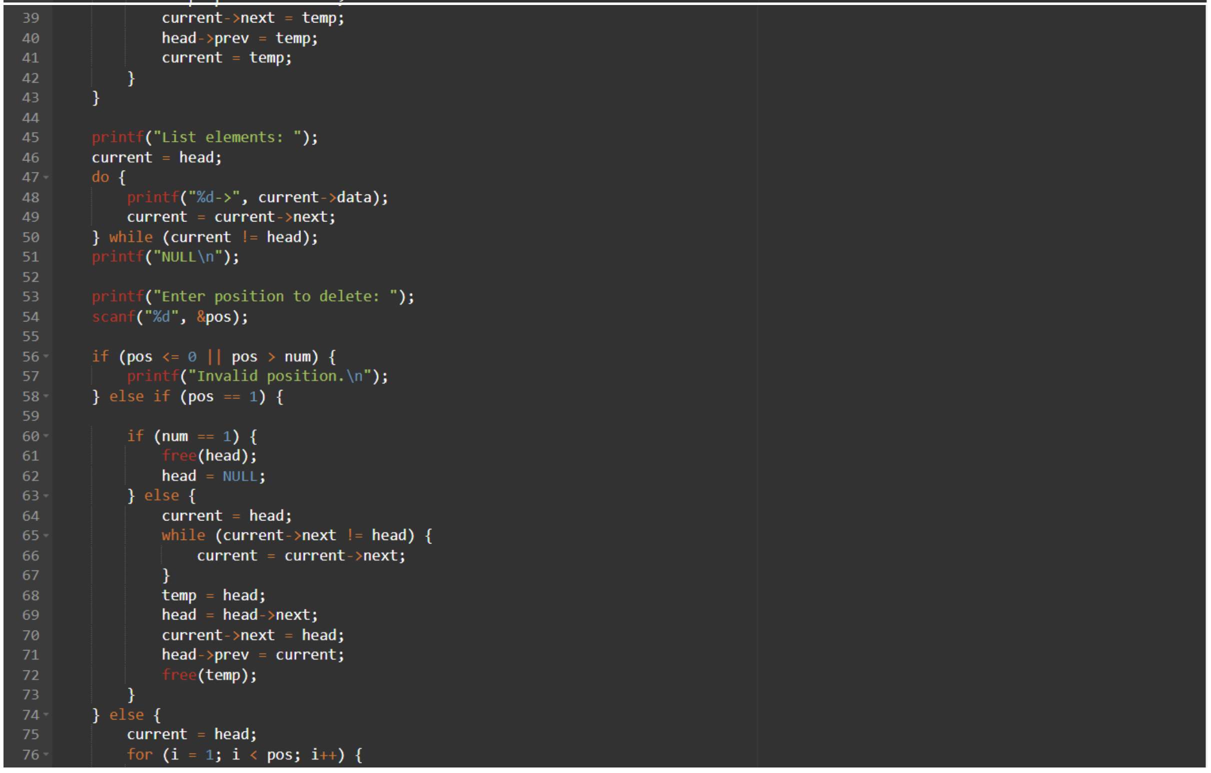
## Deletion of circular double linked list:

### Program:

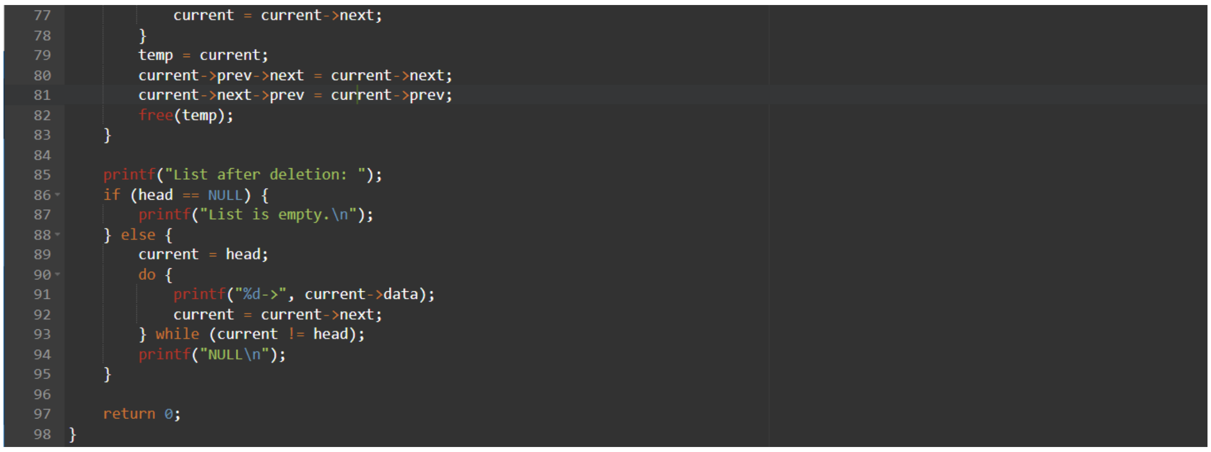


main.c

```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-5

//Deletion of circular double linked list
#include <stdio.h>
#include <stdlib.h>

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

int main() {
    struct Node* head = NULL;
    struct Node* temp = NULL;
    struct Node* current = NULL;

    int num, ele, i, pos;
    printf("Enter size of list: ");
    scanf("%d", &num);

    for (i = 0; i < num; i++) {
        printf("Enter the element: ");
        scanf("%d", &ele);

        if (head == NULL) {
            head = (struct Node*) malloc(sizeof(struct Node));
            head->data = ele;
            head->next = head;
            head->prev = head;
            current = head;
        } else {
            temp = (struct Node*) malloc(sizeof(struct Node));
            temp->data = ele;
            temp->next = head;
            temp->prev = current;
```

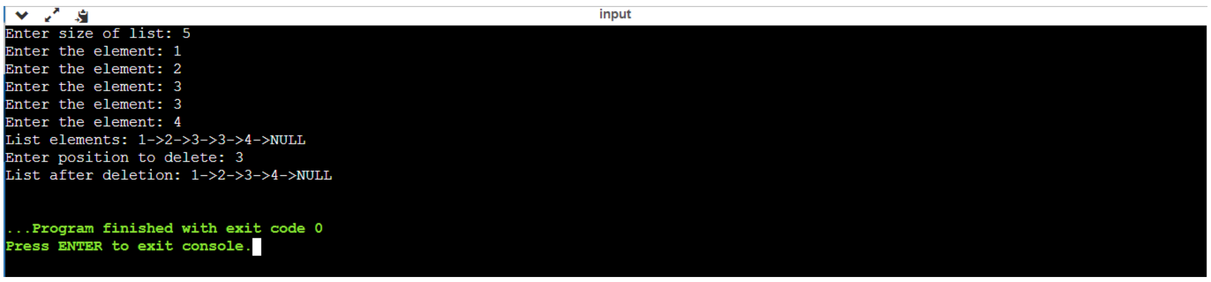


```c
            current->next = temp;
            head->prev = temp;
            current = temp;
        }
    }

    printf("List elements: ");
    current = head;
    do {
        printf("%d->", current->data);
        current = current->next;
    } while (current != head);
    printf("NULL\n");

    printf("Enter position to delete: ");
    scanf("%d", &pos);

    if (pos <= 0 || pos > num) {
        printf("Invalid position.\n");
    } else if (pos == 1) {

        if (num == 1) {
            free(head);
            head = NULL;
        } else {
            current = head;
            while (current->next != head) {
                current = current->next;
            }
            temp = head;
            head = head->next;
            current->next = head;
            head->prev = current;
            free(temp);
        }
    } else {
        current = head;
        for (i = 1; i < pos; i++) {
```



```c
            current = current->next;
        }
        temp = current;
        current->prev->next = current->next;
        current->next->prev = current->prev;
        free(temp);
    }

    printf("List after deletion: ");
    if (head == NULL) {
        printf("List is empty.\n");
    } else {
        current = head;
        do {
            printf("%d->", current->data);
            current = current->next;
        } while (current != head);
        printf("NULL\n");
    }

    return 0;
}
```

## Output:



```
Enter size of list: 5
Enter the element: 1
Enter the element: 2
Enter the element: 3
Enter the element: 3
Enter the element: 4
List elements: 1->2->3->3->4->NULL
Enter position to delete: 3
List after deletion: 1->2->3->4->NULL


...Program finished with exit code 0
Press ENTER to exit console.
```

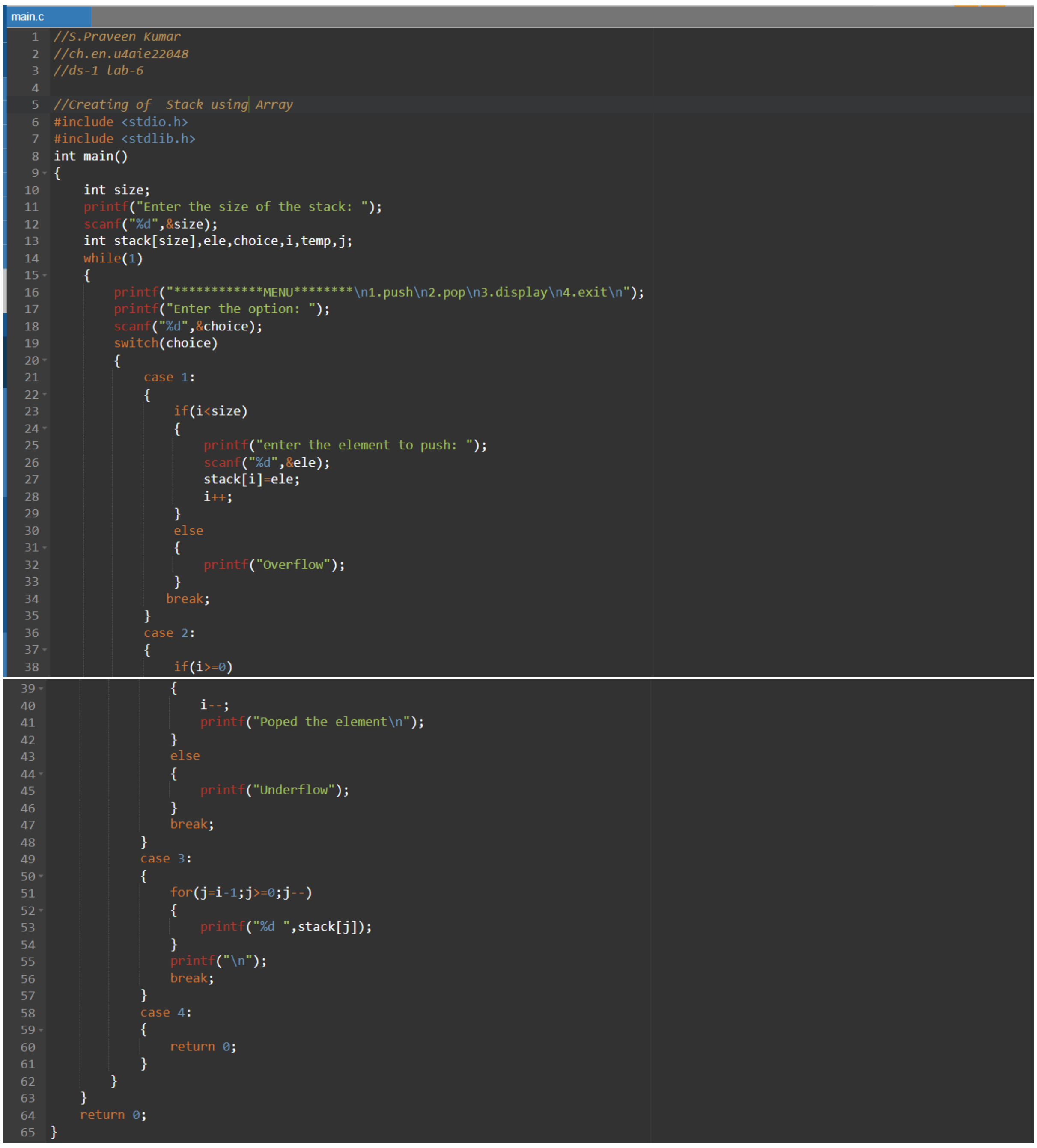
# DATA STRUCTURE – 1

## LAB-6

S.Praveen kumar
ch.en.u4aie22048

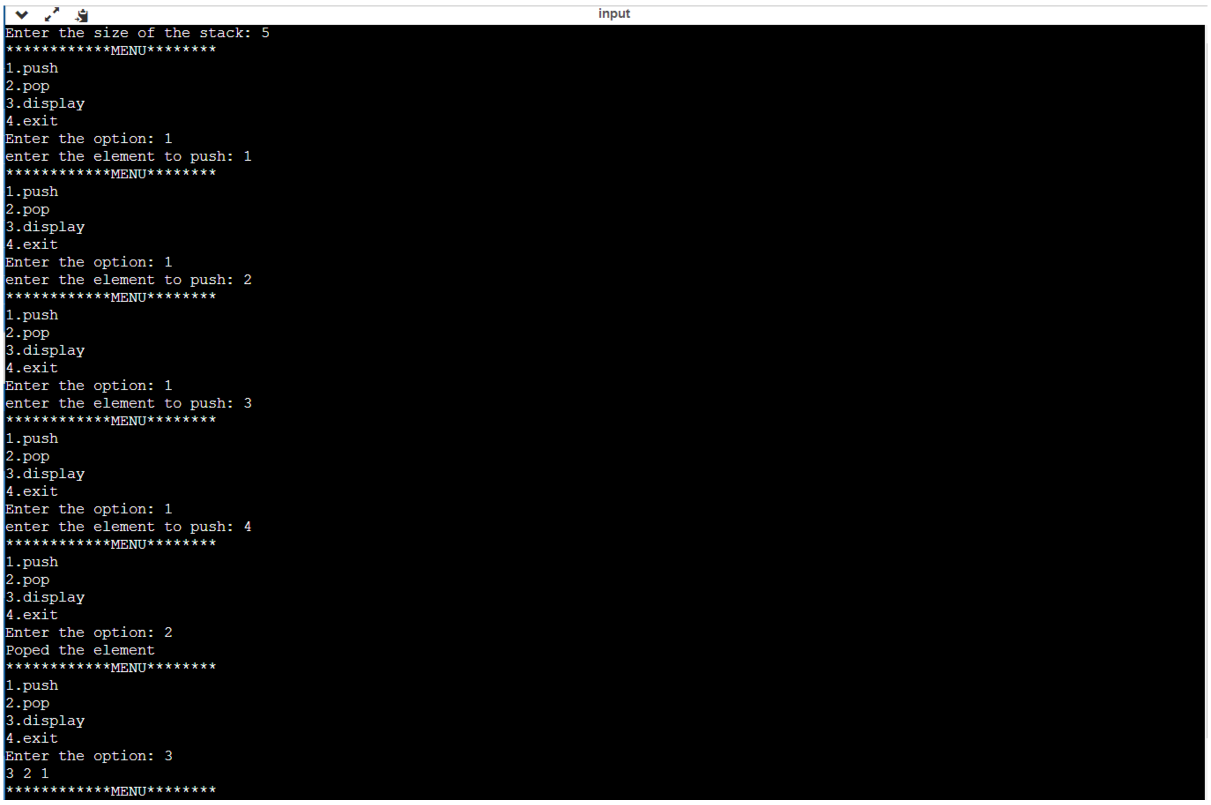
## Creation of stack using Array:

### Program:



```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-6

//Creating of  Stack using Array
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int size;
    printf("Enter the size of the stack: ");
    scanf("%d",&size);
    int stack[size],ele,choice,i,temp,j;
    while(1)
    {
        printf("************MENU********\n1.push\n2.pop\n3.display\n4.exit\n");
        printf("Enter the option: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                if(i<size)
                {
                    printf("enter the element to push: ");
                    scanf("%d",&ele);
                    stack[i]=ele;
                    i++;
                }
                else
                {
                    printf("Overflow");
                }
               break;
            }
            case 2:
            {
                if(i>=0)
                {
                    i--;
                    printf("Poped the element\n");
                }
                else
                {
                    printf("Underflow");
                }
                break;
            }
            case 3:
            {
                for(j=i-1;j>=0;j--)
                {
                    printf("%d ",stack[j]);
                }
                printf("\n");
                break;
            }
            case 4:
            {
                return 0;
            }
        }
    }
    return 0;
}
```

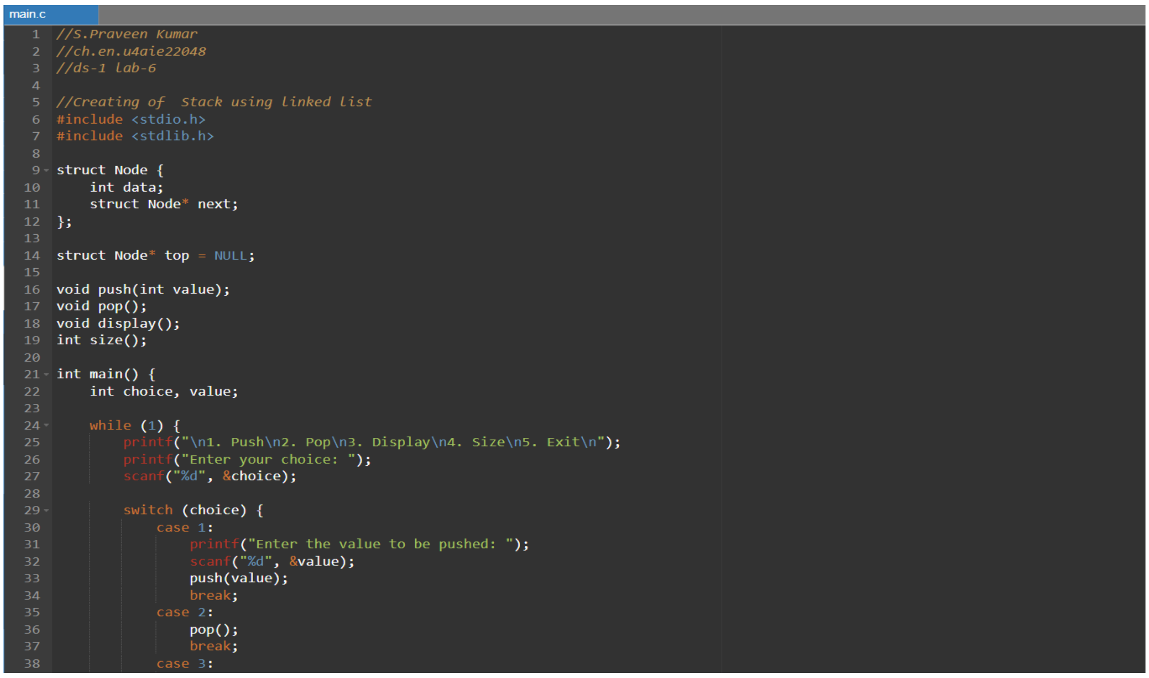
## Output:



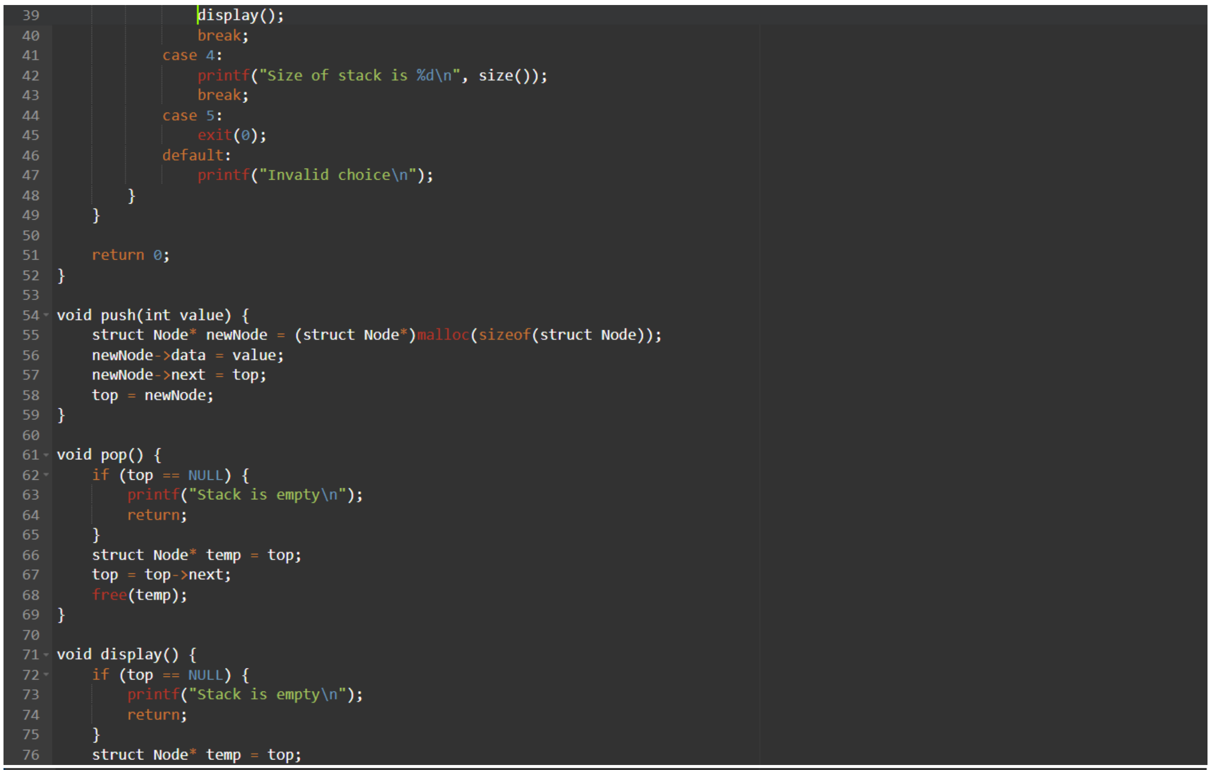
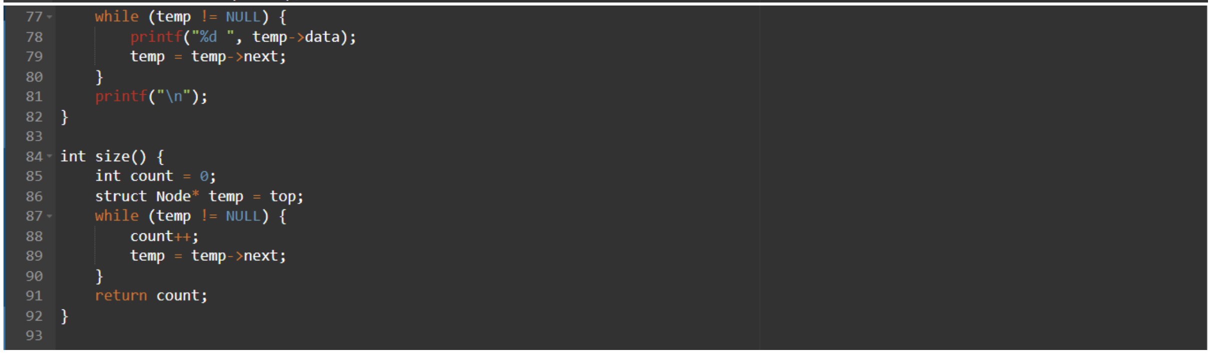
```
Enter the size of the stack: 5
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 1
enter the element to push: 1
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 1
enter the element to push: 2
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 1
enter the element to push: 3
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 1
enter the element to push: 4
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 2
Poped the element
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 3
3 2 1
************MENU********
```

## Creation of Stack using linked list:

### Program:

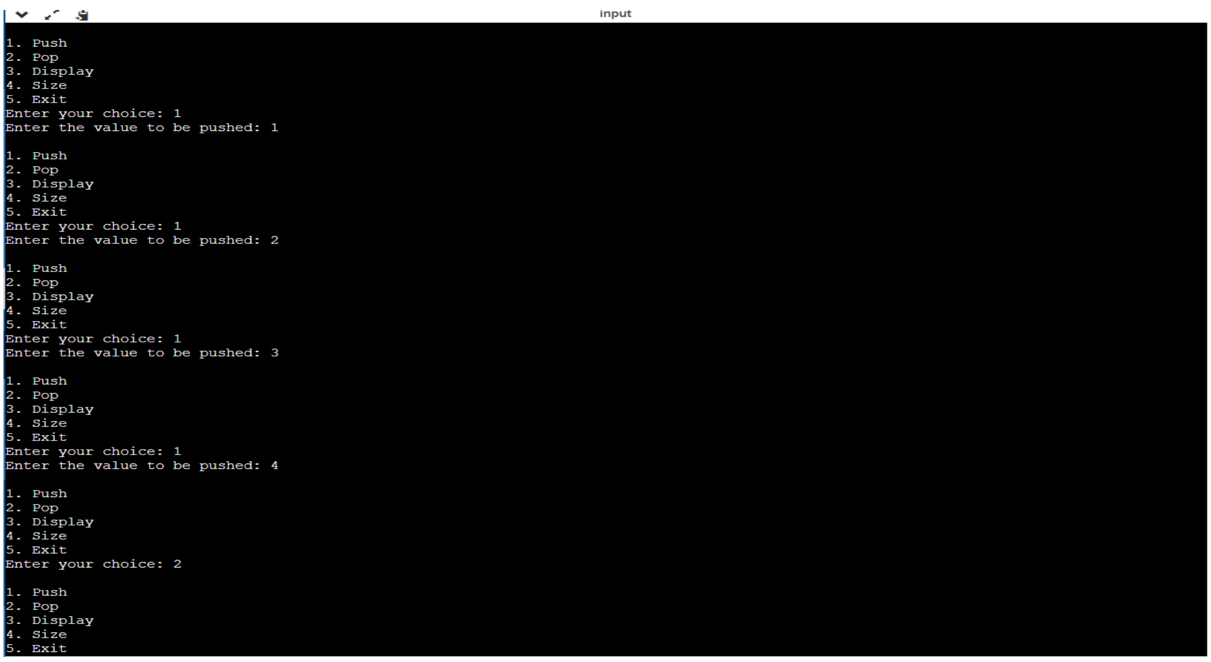


```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-6

//Creating of  Stack using linked list
#include <stdio.h>
#include <stdlib.h>

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

struct Node* top = NULL;

void push(int value);
void pop();
void display();
int size();

int main() {
    int choice, value;

    while (1) {
        printf("\n1. Push\n2. Pop\n3. Display\n4. Size\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the value to be pushed: ");
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
```

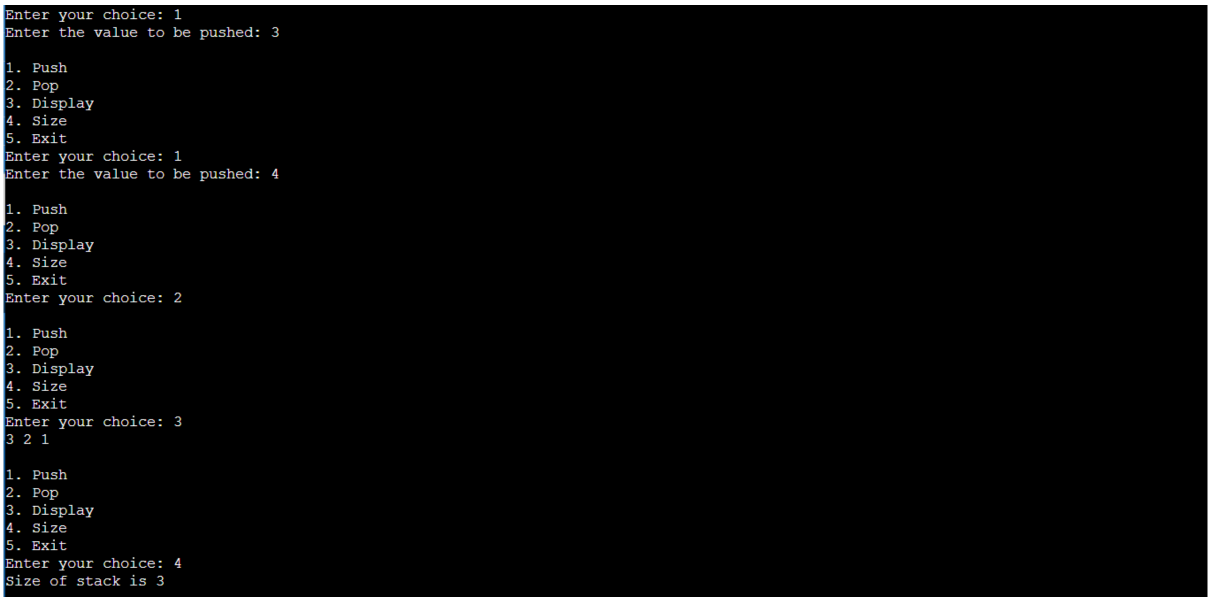



```c
                display();
                break;
            case 4:
                printf("Size of stack is %d\n", size());
                break;
            case 5:
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    }

    return 0;
}

void push(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = top;
    top = newNode;
}

void pop() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    struct Node* temp = top;
    top = top->next;
    free(temp);
}

void display() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    struct Node* temp = top;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int size() {
    int count = 0;
    struct Node* temp = top;
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }
    return count;
}
```

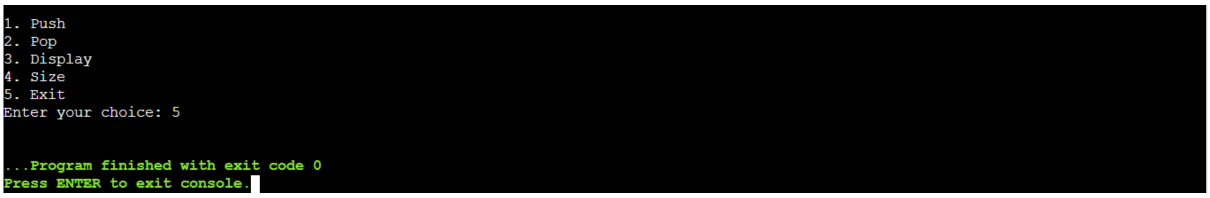
## Output:



```
1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 1

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 2

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 3

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 4

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 2

1. Push
2. Pop
3. Display
4. Size
5. Exit
```



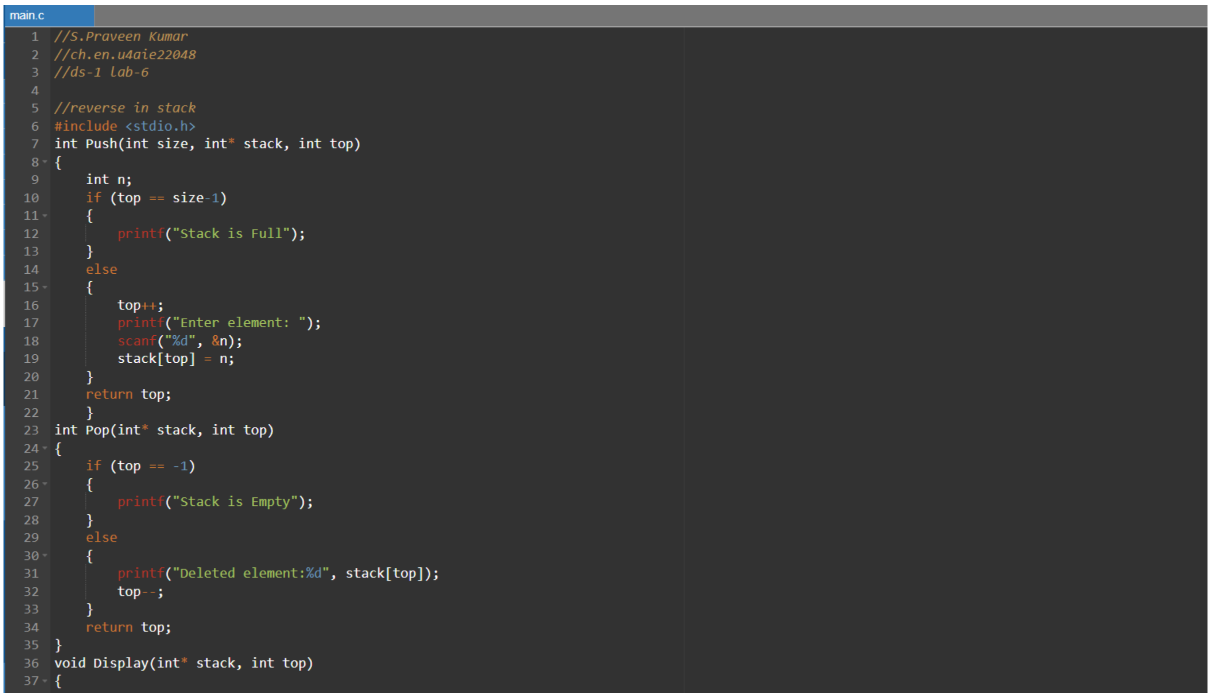
```
Enter your choice: 1
Enter the value to be pushed: 3

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 1
Enter the value to be pushed: 4

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 2

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 3
3 2 1

1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 4
Size of stack is 3
```



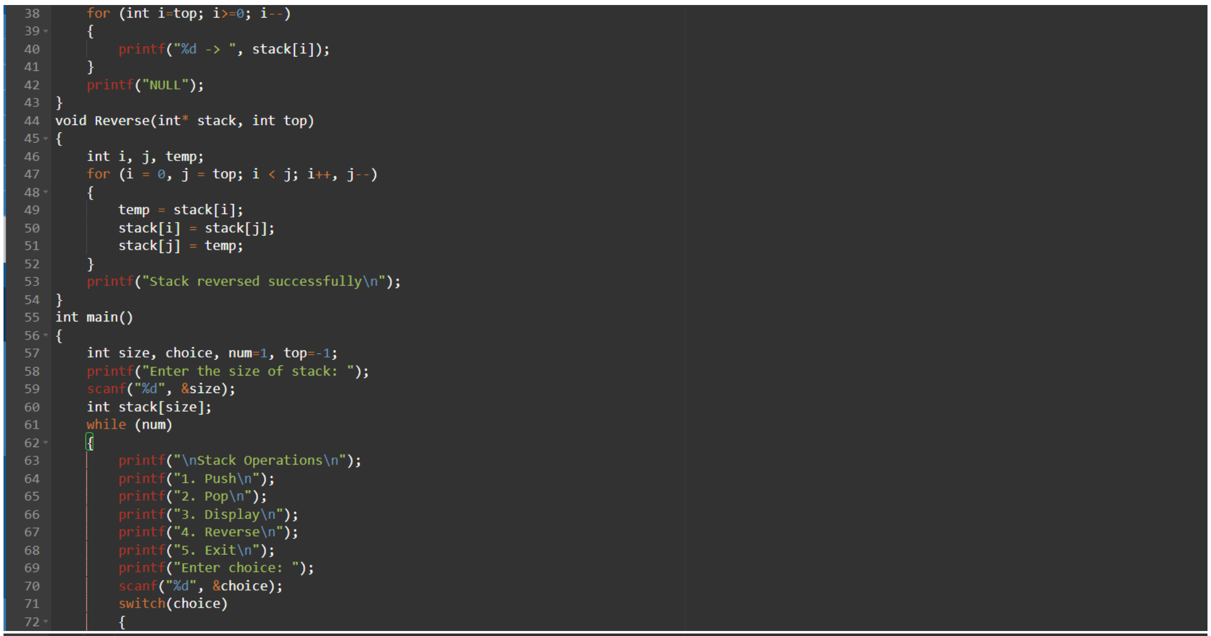
```
1. Push
2. Pop
3. Display
4. Size
5. Exit
Enter your choice: 5


...Program finished with exit code 0
Press ENTER to exit console.
```

## Reversing of Stack using linked list:

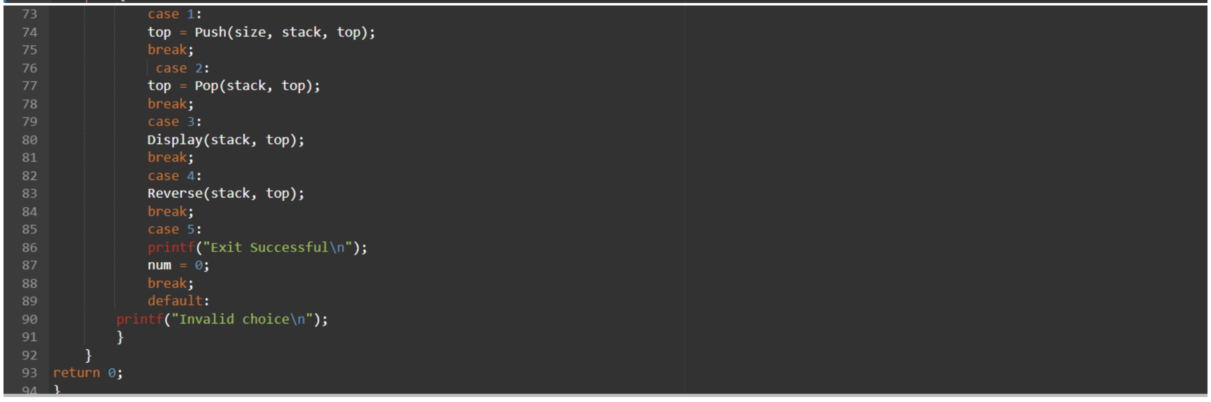
### Program:



```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-6

//reverse in stack
#include <stdio.h>
int Push(int size, int* stack, int top)
{
    int n;
    if (top == size-1)
    {
        printf("Stack is Full");
    }
    else
    {
        top++;
        printf("Enter element: ");
        scanf("%d", &n);
        stack[top] = n;
    }
    return top;
}
int Pop(int* stack, int top)
{
    if (top == -1)
    {
        printf("Stack is Empty");
    }
    else
    {
        printf("Deleted element:%d", stack[top]);
        top--;
    }
    return top;
}
void Display(int* stack, int top)
{
```

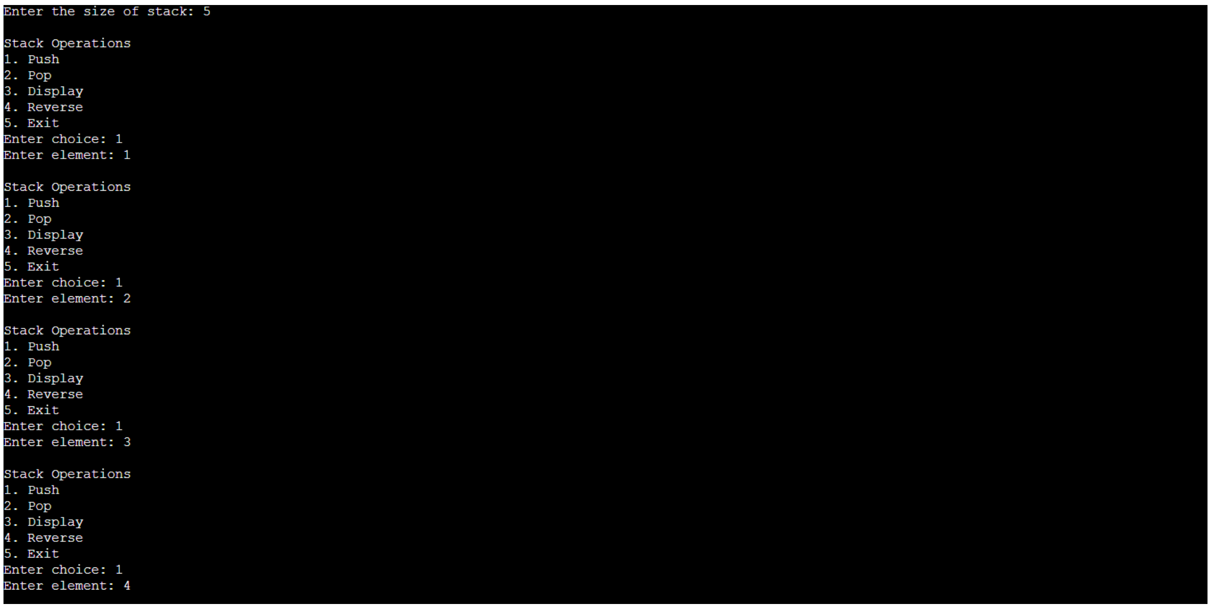


```c
    for (int i=top; i>=0; i--)
    {
        printf("%d -> ", stack[i]);
    }
    printf("NULL");
}
void Reverse(int* stack, int top)
{
    int i, j, temp;
    for (i = 0, j = top; i < j; i++, j--)
    {
        temp = stack[i];
        stack[i] = stack[j];
        stack[j] = temp;
    }
    printf("Stack reversed successfully\n");
}
int main()
{
    int size, choice, num=1, top=-1;
    printf("Enter the size of stack: ");
    scanf("%d", &size);
    int stack[size];
    while (num)
    {
        printf("\nStack Operations\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Reverse\n");
        printf("5. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch(choice)
        {
```

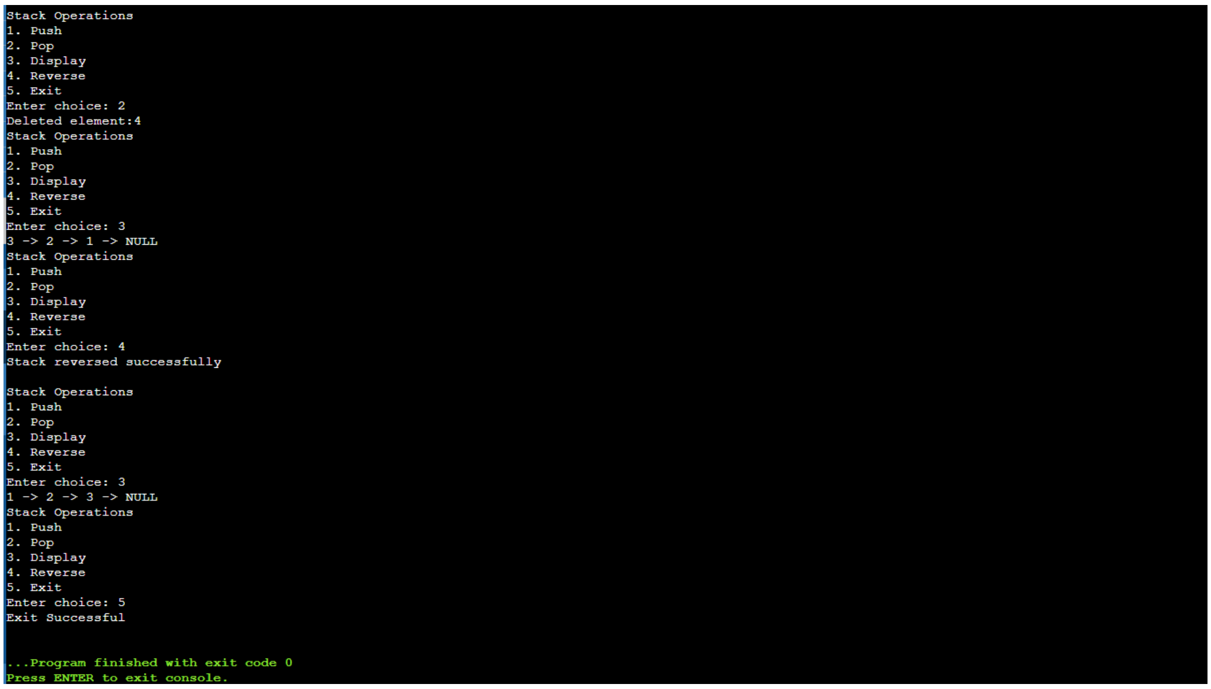


```c
            case 1:
            top = Push(size, stack, top);
            break;
             case 2:
            top = Pop(stack, top);
            break;
            case 3:
            Display(stack, top);
            break;
            case 4:
            Reverse(stack, top);
            break;
            case 5:
            printf("Exit Successful\n");
            num = 0;
            break;
            default:
        printf("Invalid choice\n");
        }
    }
return 0;
}
```

## Output:



```
Enter the size of stack: 5

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 1
Enter element: 1

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 1
Enter element: 2

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 1
Enter element: 3

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 1
Enter element: 4
```



```
Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 2
Deleted element:4
Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 3
3 -> 2 -> 1 -> NULL
Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 4
Stack reversed successfully

Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 3
1 -> 2 -> 3 -> NULL
Stack Operations
1. Push
2. Pop
3. Display
4. Reverse
5. Exit
Enter choice: 5
Exit Successful


...Program finished with exit code 0
Press ENTER to exit console.
```

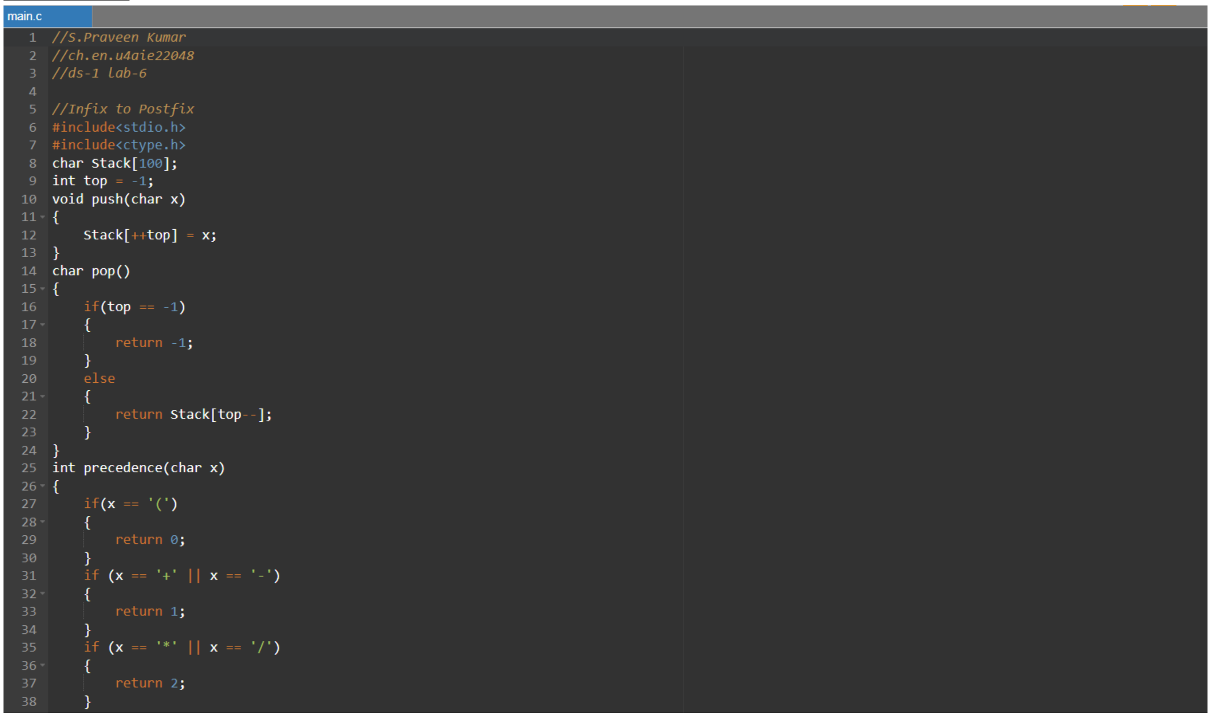
# DATA STRUCTURE – 1

# LAB-7

S.Praveen kumar
ch.en.u4aie22048

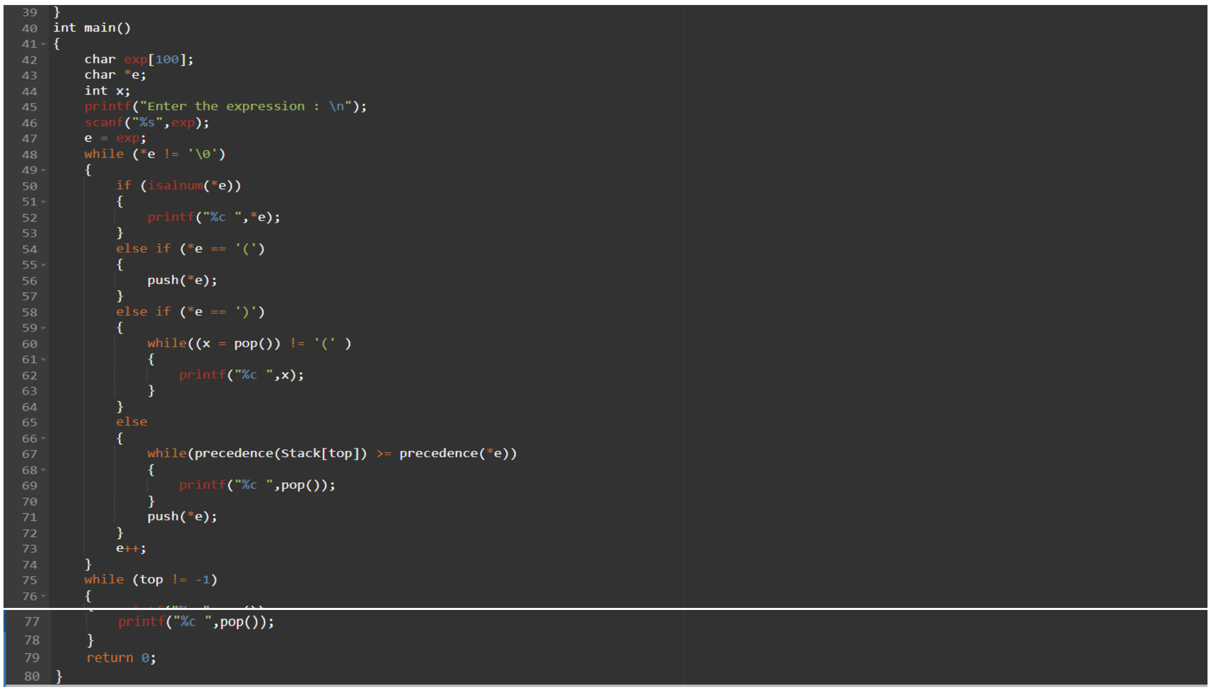
## Infix to Postfix

### Program:



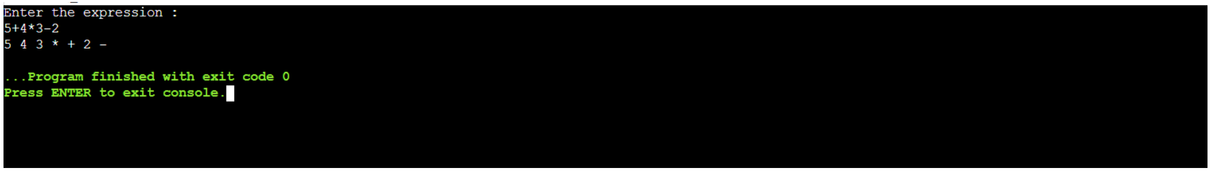
main.c

```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-6

//Infix to Postfix
#include<stdio.h>
#include<ctype.h>
char Stack[100];
int top = -1;
void push(char x)
{
    Stack[++top] = x;
}
char pop()
{
    if(top == -1)
    {
        return -1;
    }
    else
    {
        return Stack[top--];
    }
}
int precedence(char x)
{
    if(x == '(')
    {
        return 0;
    }
    if (x == '+' || x == '-')
    {
        return 1;
    }
    if (x == '*' || x == '/')
    {
        return 2;
    }
```



```c
}
int main()
{
    char exp[100];
    char *e;
    int x;
    printf("Enter the expression : \n");
    scanf("%s",exp);
    e = exp;
    while (*e != '\0')
    {
        if (isalnum(*e))
        {
            printf("%c ",*e);
        }
        else if (*e == '(')
        {
            push(*e);
        }
        else if (*e == ')')
        {
            while((x = pop()) != '(' )
            {
                printf("%c ",x);
            }
        }
        else
        {
            while(precedence(Stack[top]) >= precedence(*e))
            {
                printf("%c ",pop());
            }
            push(*e);
        }
        e++;
    }
    while (top != -1)
    {
        printf("%c ",pop());
    }
    return 0;
}
```

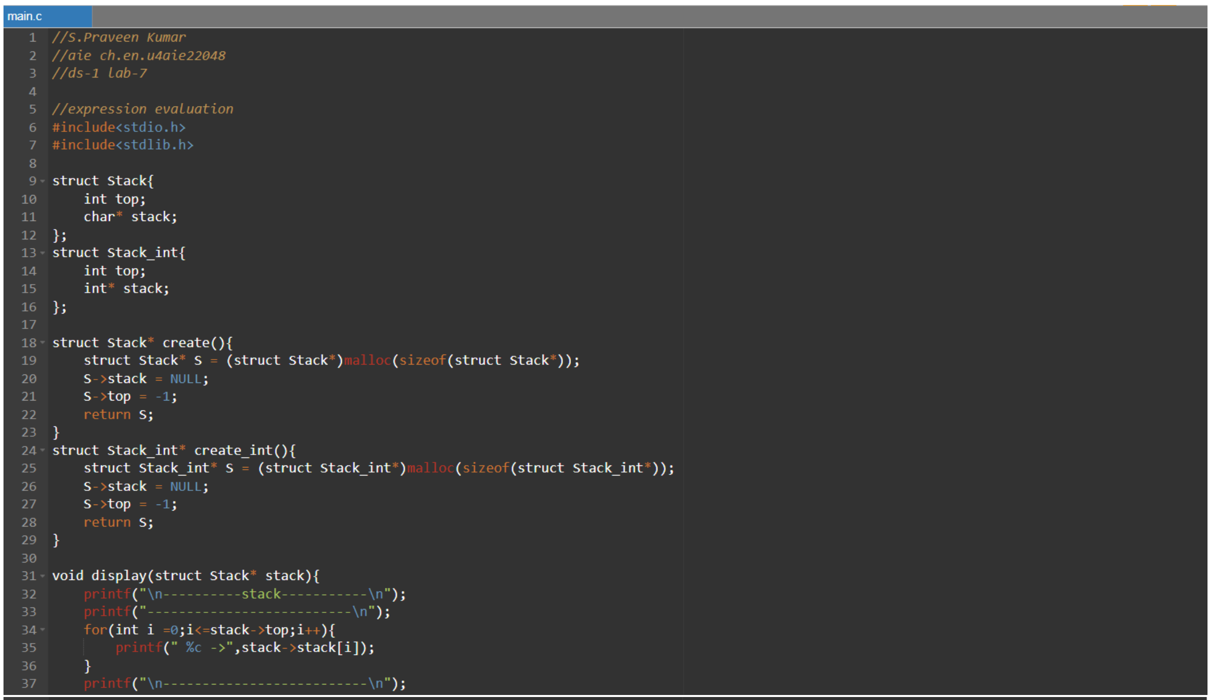
## Output:



```
Enter the expression :
5+4*3-2
5 4 3 * + 2 -

...Program finished with exit code 0
Press ENTER to exit console.
```

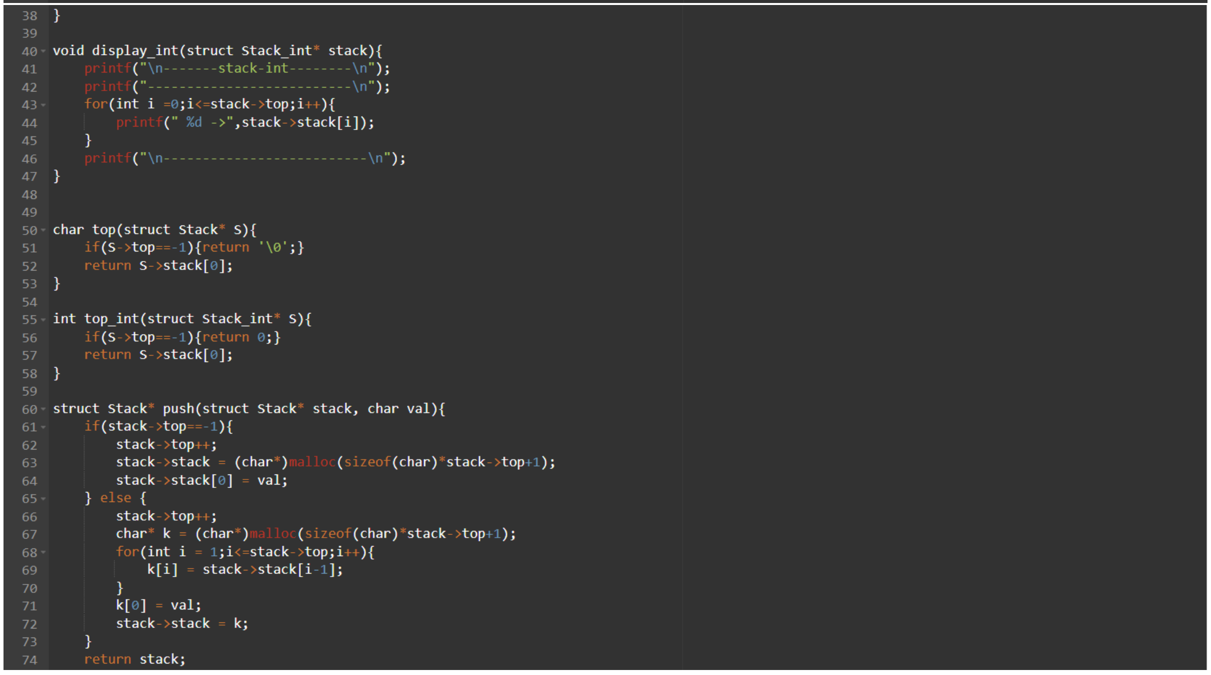
## Expression Evaluation

## Program:

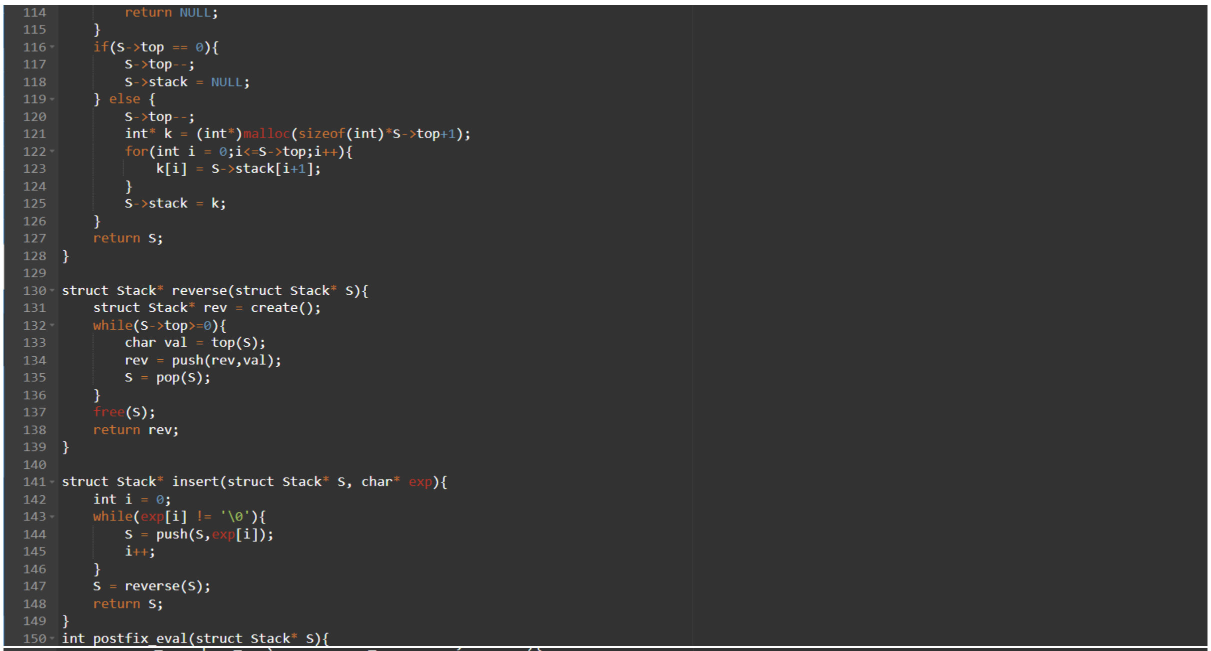


main.c

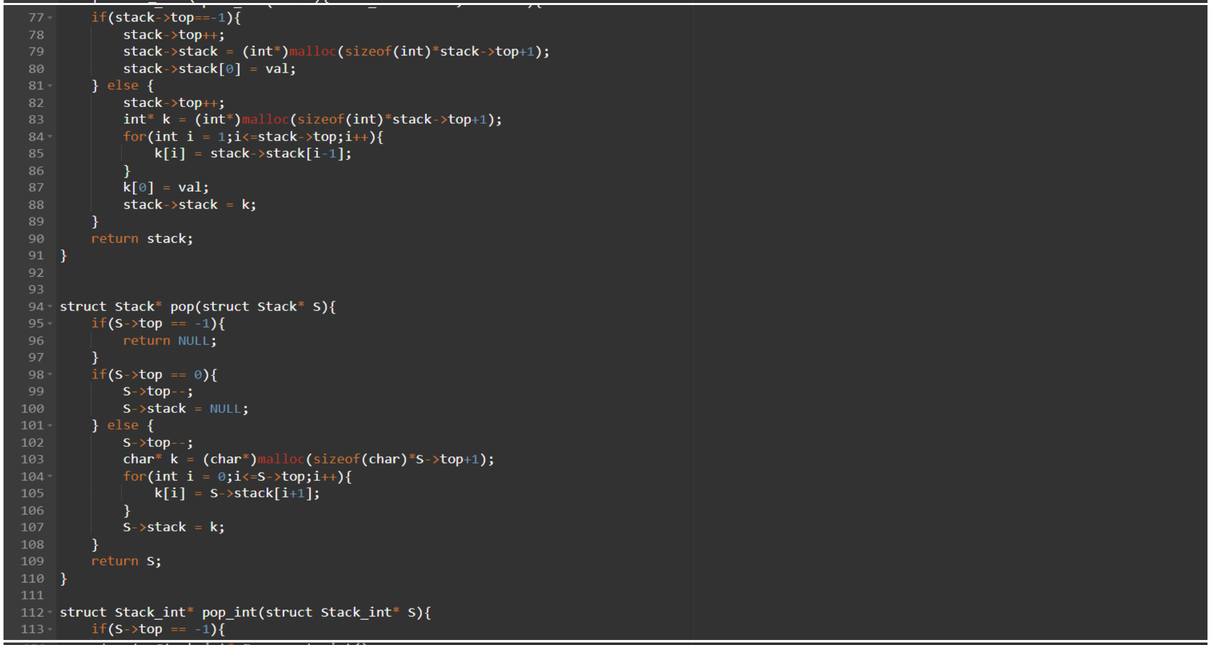
```c
//S.Praveen Kumar
//aie ch.en.u4aie22048
//ds-1 lab-7

//expression evaluation
#include<stdio.h>
#include<stdlib.h>

struct Stack{
    int top;
    char* stack;
};
struct Stack_int{
    int top;
    int* stack;
};

struct Stack* create(){
    struct Stack* S = (struct Stack*)malloc(sizeof(struct Stack*));
    S->stack = NULL;
    S->top = -1;
    return S;
}
struct Stack_int* create_int(){
    struct Stack_int* S = (struct Stack_int*)malloc(sizeof(struct Stack_int*));
    S->stack = NULL;
    S->top = -1;
    return S;
}

void display(struct Stack* stack){
    printf("\n---------stack----------\n");
    printf("------------------------\n");
    for(int i =0;i<=stack->top;i++){
        printf(" %c ->",stack->stack[i]);
    }
    printf("\n------------------------\n");
```



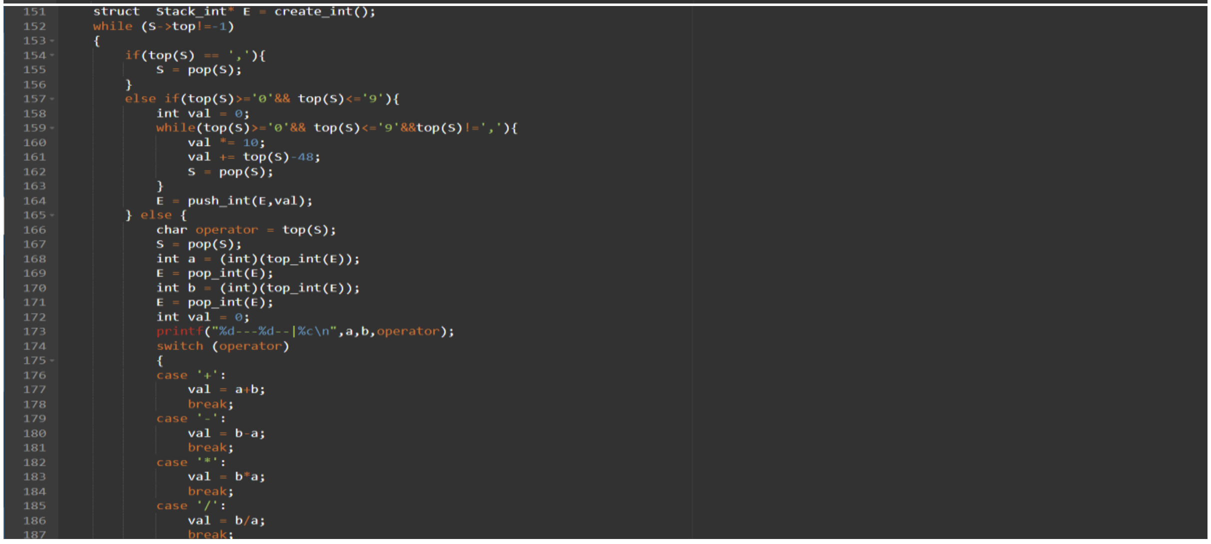
```c
}

void display_int(struct Stack_int* stack){
    printf("\n-------stack-int--------\n");
    printf("------------------------\n");
    for(int i =0;i<=stack->top;i++){
        printf(" %d ->",stack->stack[i]);
    }
    printf("\n------------------------\n");
}


char top(struct Stack* S){
    if(S->top==-1){return '\0';}
    return S->stack[0];
}

int top_int(struct Stack_int* S){
    if(S->top==-1){return 0;}
    return S->stack[0];
}

struct Stack* push(struct Stack* stack, char val){
    if(stack->top==-1){
        stack->top++;
        stack->stack = (char*)malloc(sizeof(char)*stack->top+1);
        stack->stack[0] = val;
    } else {
        stack->top++;
        char* k = (char*)malloc(sizeof(char)*stack->top+1);
        for(int i = 1;i<=stack->top;i++){
            k[i] = stack->stack[i-1];
        }
        k[0] = val;
        stack->stack = k;
    }
    return stack;
```



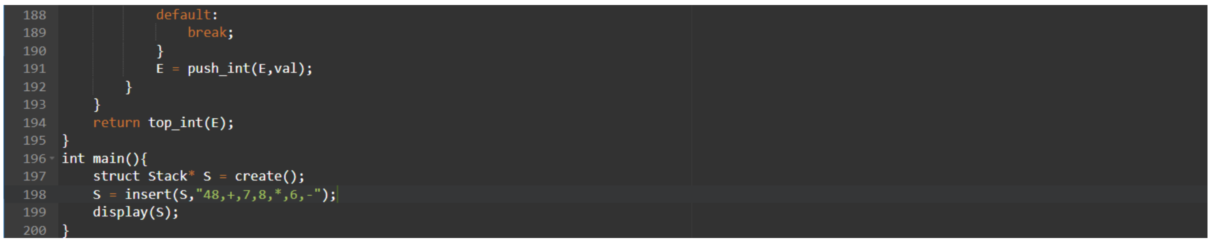
```c
        return NULL;
    }
    if(S->top == 0){
        S->top--;
        S->stack = NULL;
    } else {
        S->top--;
        int* k = (int*)malloc(sizeof(int)*S->top+1);
        for(int i = 0;i<=S->top;i++){
            k[i] = S->stack[i+1];
        }
        S->stack = k;
    }
    return S;
}

struct Stack* reverse(struct Stack* S){
    struct Stack* rev = create();
    while(S->top>=0){
        char val = top(S);
        rev = push(rev,val);
        S = pop(S);
    }
    free(S);
    return rev;
}

struct Stack* insert(struct Stack* S, char* exp){
    int i = 0;
    while(exp[i] != '\0'){
        S = push(S,exp[i]);
        i++;
    }
    S = reverse(S);
    return S;
}
int postfix_eval(struct Stack* S){
```



```c
    if(stack->top==-1){
        stack->top++;
        stack->stack = (int*)malloc(sizeof(int)*stack->top+1);
        stack->stack[0] = val;
    } else {
        stack->top++;
        int* k = (int*)malloc(sizeof(int)*stack->top+1);
        for(int i = 1;i<=stack->top;i++){
            k[i] = stack->stack[i-1];
        }
        k[0] = val;
        stack->stack = k;
    }
    return stack;
}


struct Stack* pop(struct Stack* S){
    if(S->top == -1){
        return NULL;
    }
    if(S->top == 0){
        S->top--;
        S->stack = NULL;
    } else {
        S->top--;
        char* k = (char*)malloc(sizeof(char)*S->top+1);
        for(int i = 0;i<=S->top;i++){
            k[i] = S->stack[i+1];
        }
        S->stack = k;
    }
    return S;
}

struct Stack_int* pop_int(struct Stack_int* S){
    if(S->top == -1){
```

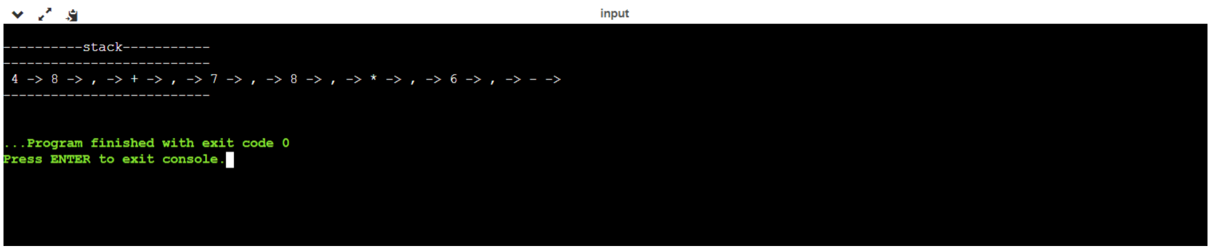


```c
    struct  Stack_int* E = create_int();
    while (S->top!=-1)
    {
        if(top(S) == ','){
            S = pop(S);
        }
        else if(top(S)>='0'&& top(S)<='9'){
            int val = 0;
            while(top(S)>='0'&& top(S)<='9'&&top(S)!=','){
                val *= 10;
                val += top(S)-48;
                S = pop(S);
            }
            E = push_int(E,val);
        } else {
            char operator = top(S);
            S = pop(S);
            int a = (int)(top_int(E));
            E = pop_int(E);
            int b = (int)(top_int(E));
            E = pop_int(E);
            int val = 0;
            printf("%d---%d--|%c\n",a,b,operator);
            switch (operator)
            {
            case '+':
                val = a+b;
                break;
            case '-':
                val = b-a;
                break;
            case '*':
                val = b*a;
                break;
            case '/':
                val = b/a;
                break;
```



```c
            default:
                break;
            }
            E = push_int(E,val);
        }
    }
    return top_int(E);
}
int main(){
    struct Stack* S = create();
    S = insert(S,"48,+,7,8,*,6,-");
    display(S);
}
```

## Output:



```
----------stack-----------
--------------------------
 4 -> 8 -> , -> + -> , -> 7 -> , -> 8 -> , -> * -> , -> 6 -> , -> - ->
--------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```

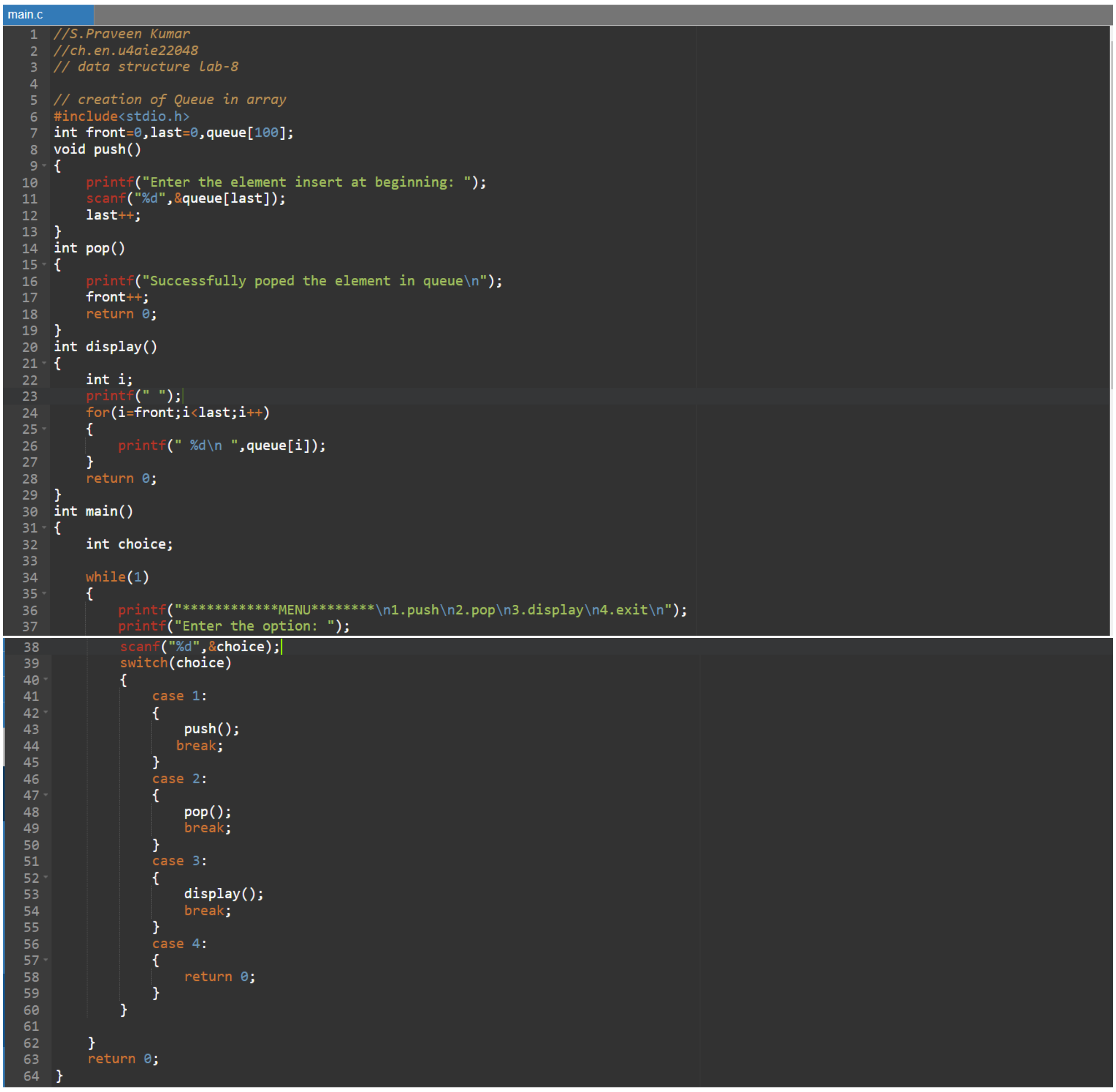
# DATA STRUCTURE – 1

# LAB-8

S.Praveen kumar
ch.en.u4aie22048

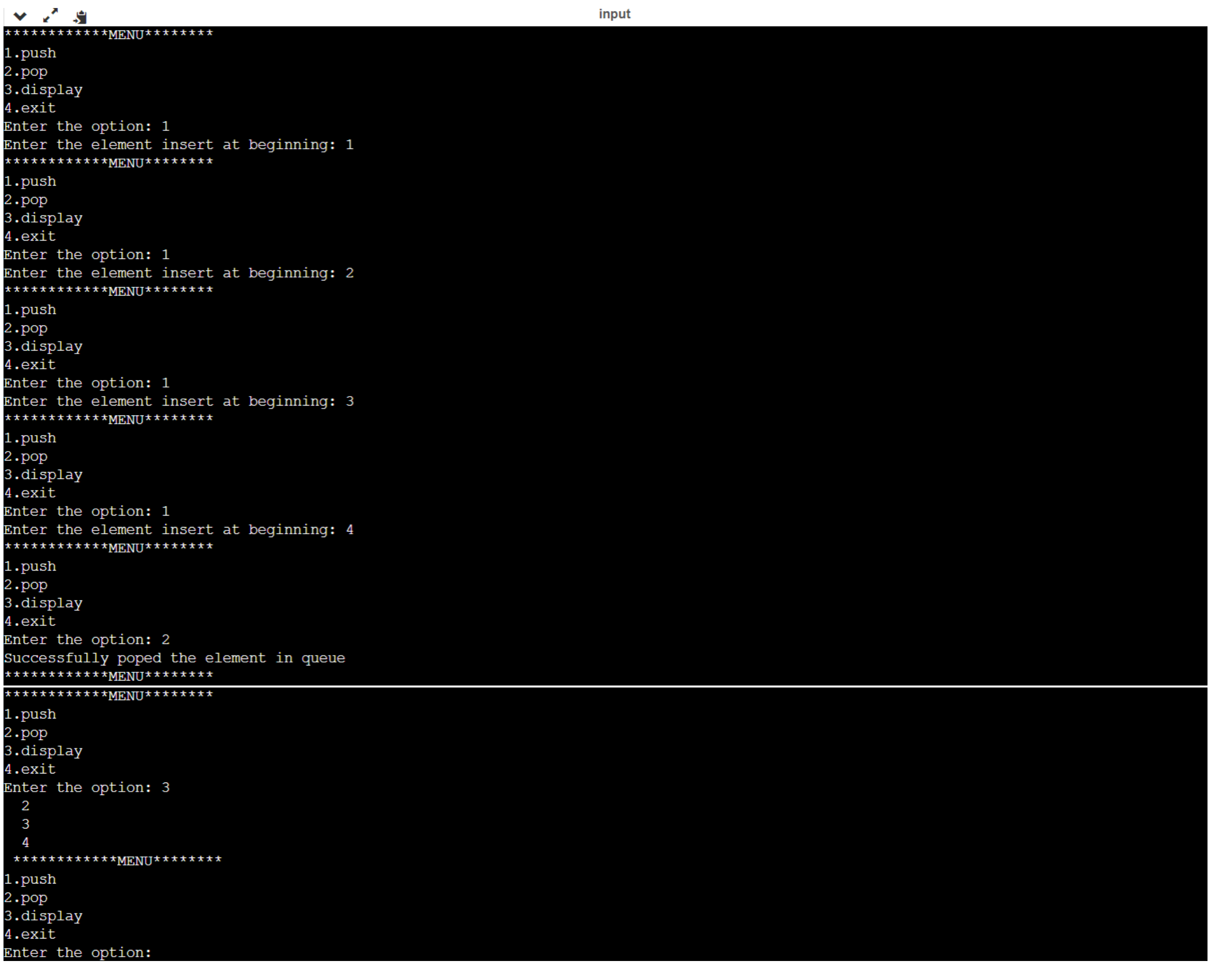
## Creation of Queue in array:

### Program:



```c
//S.Praveen Kumar
//ch.en.u4aie22048
// data structure lab-8

// creation of Queue in array
#include<stdio.h>
int front=0,last=0,queue[100];
void push()
{
    printf("Enter the element insert at beginning: ");
    scanf("%d",&queue[last]);
    last++;
}
int pop()
{
    printf("Successfully poped the element in queue\n");
    front++;
    return 0;
}
int display()
{
    int i;
    printf(" ");
    for(i=front;i<last;i++)
    {
        printf(" %d\n ",queue[i]);
    }
    return 0;
}
int main()
{
    int choice;

    while(1)
    {
        printf("************MENU********\n1.push\n2.pop\n3.display\n4.exit\n");
        printf("Enter the option: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
               break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            case 4:
            {
                return 0;
            }
        }

    }
    return 0;
}
```

## Output:

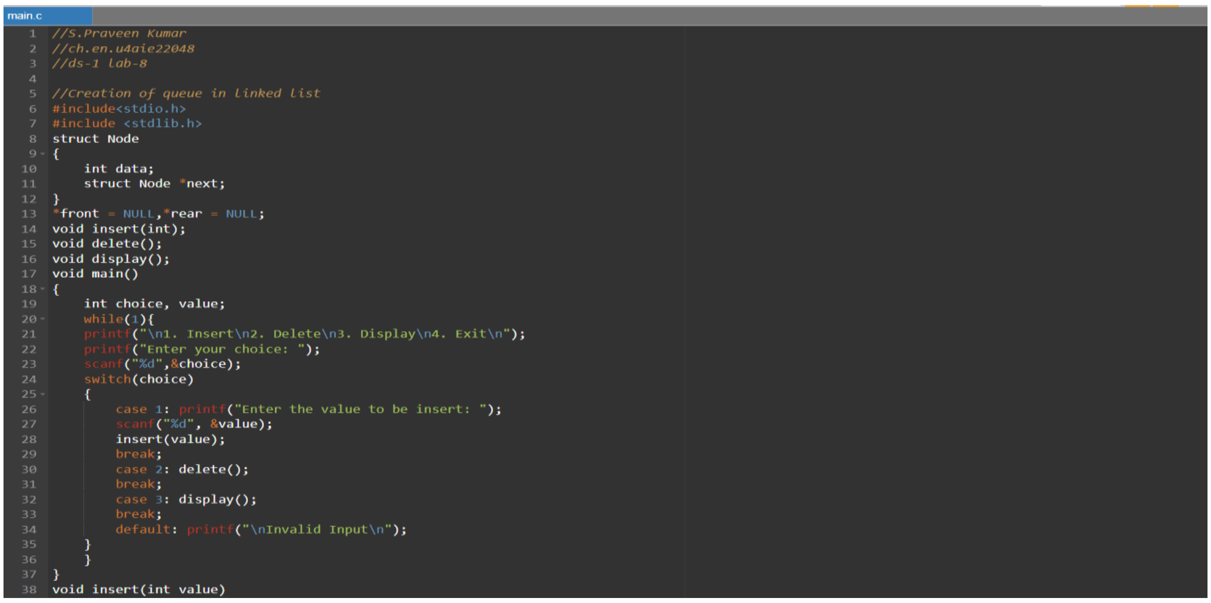


```
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 1
Enter the element insert at beginning: 1
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 1
Enter the element insert at beginning: 2
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 1
Enter the element insert at beginning: 3
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 1
Enter the element insert at beginning: 4
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 2
Successfully poped the element in queue
************MENU********
```

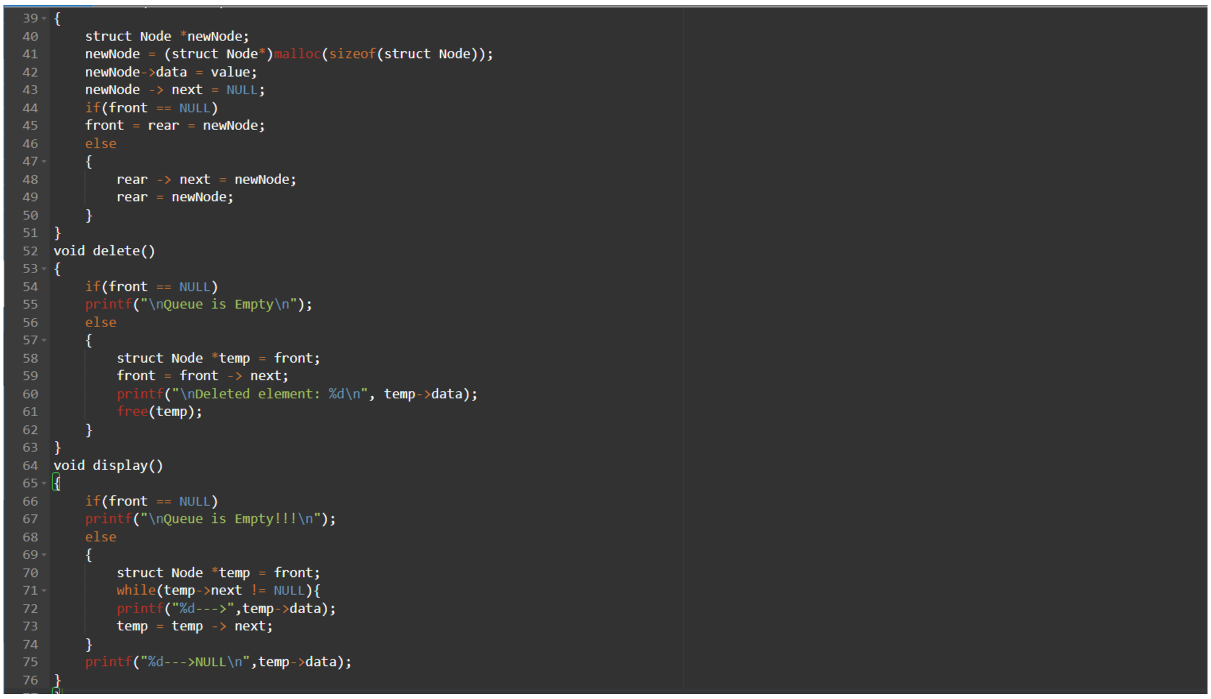
```
************MENU********
1.push
2.pop
3.display
4.exit
Enter the option: 3
  2
  3
  4
 ************MENU********
1.push
2.pop
3.display
4.exit
Enter the option:
```

## Creation of Queue in Linked List:

## Program:

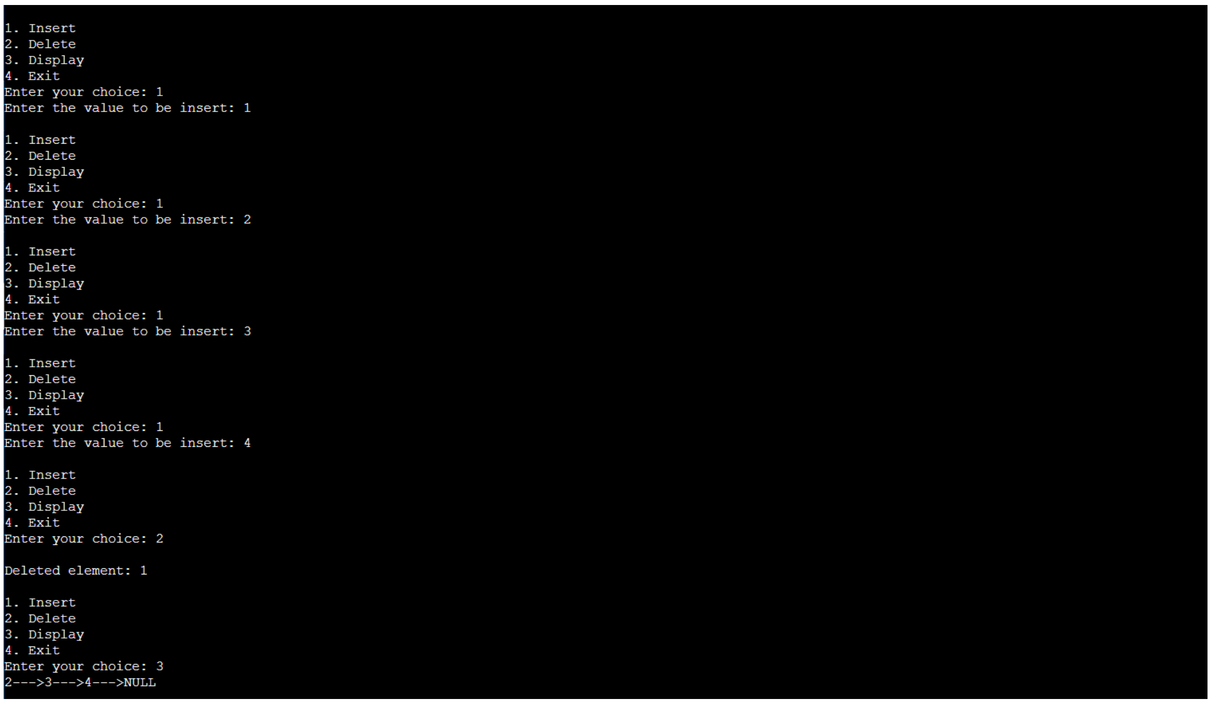


```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-8

//Creation of queue in linked list
#include<stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node *next;
}
*front = NULL,*rear = NULL;
void insert(int);
void delete();
void display();
void main()
{
    int choice, value;
    while(1){
    printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("Enter the value to be insert: ");
        scanf("%d", &value);
        insert(value);
        break;
        case 2: delete();
        break;
        case 3: display();
        break;
        default: printf("\nInvalid Input\n");
    }
    }
}
void insert(int value)
```



```c
{
    struct Node *newNode;
    newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode -> next = NULL;
    if(front == NULL)
    front = rear = newNode;
    else
    {
        rear -> next = newNode;
        rear = newNode;
    }
}
void delete()
{
    if(front == NULL)
    printf("\nQueue is Empty\n");
    else
    {
        struct Node *temp = front;
        front = front -> next;
        printf("\nDeleted element: %d\n", temp->data);
        free(temp);
    }
}
void display()
{
    if(front == NULL)
    printf("\nQueue is Empty!!!\n");
    else
    {
        struct Node *temp = front;
        while(temp->next != NULL){
        printf("%d--->",temp->data);
        temp = temp -> next;
    }
    printf("%d--->NULL\n",temp->data);
}
```

## Output:



```
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 1

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 2

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 3

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the value to be insert: 4

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2

Deleted element: 1

1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
2--->3--->4--->NULL
```

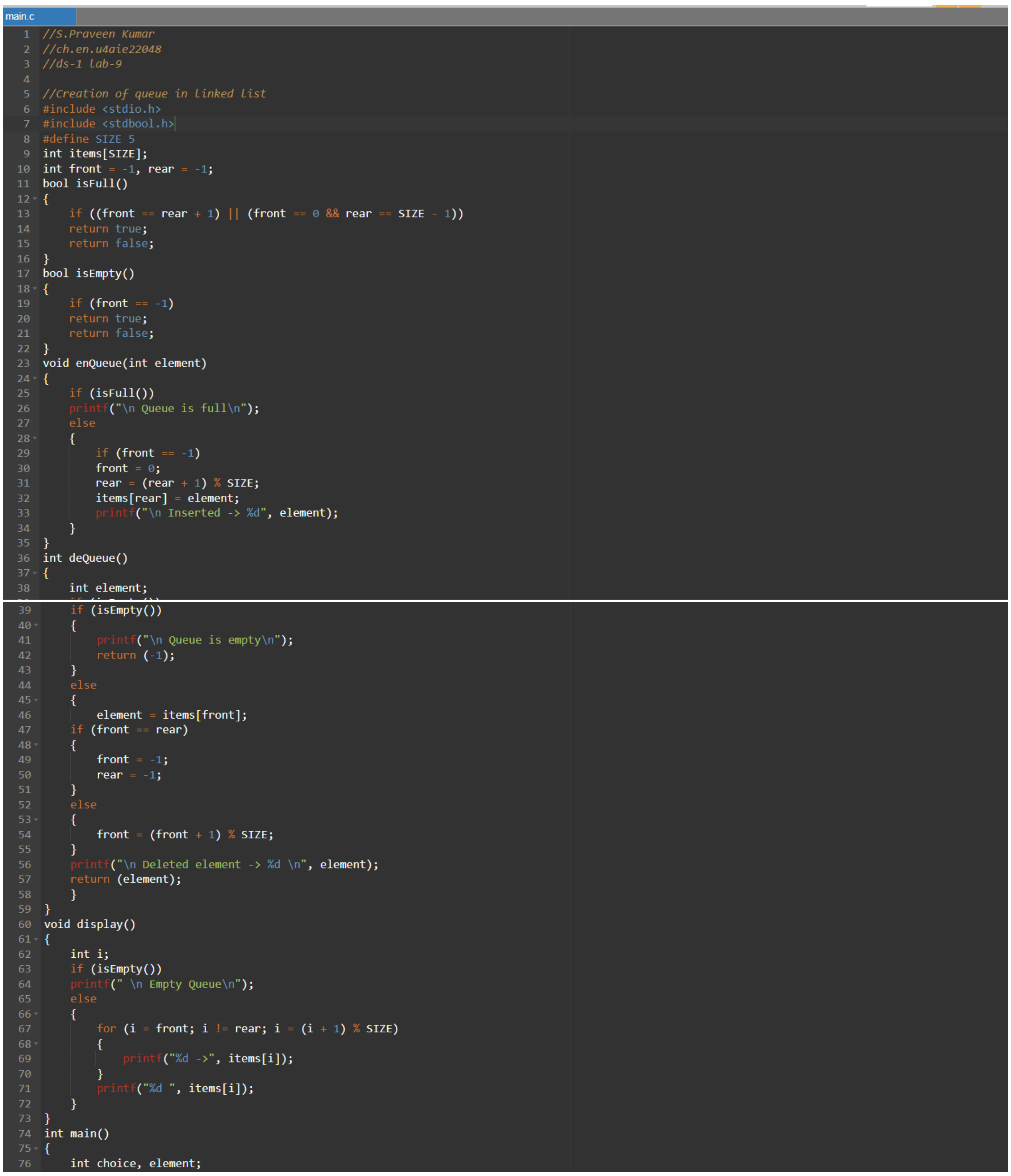
# DATA STRUCTURE – 1

# LAB-9

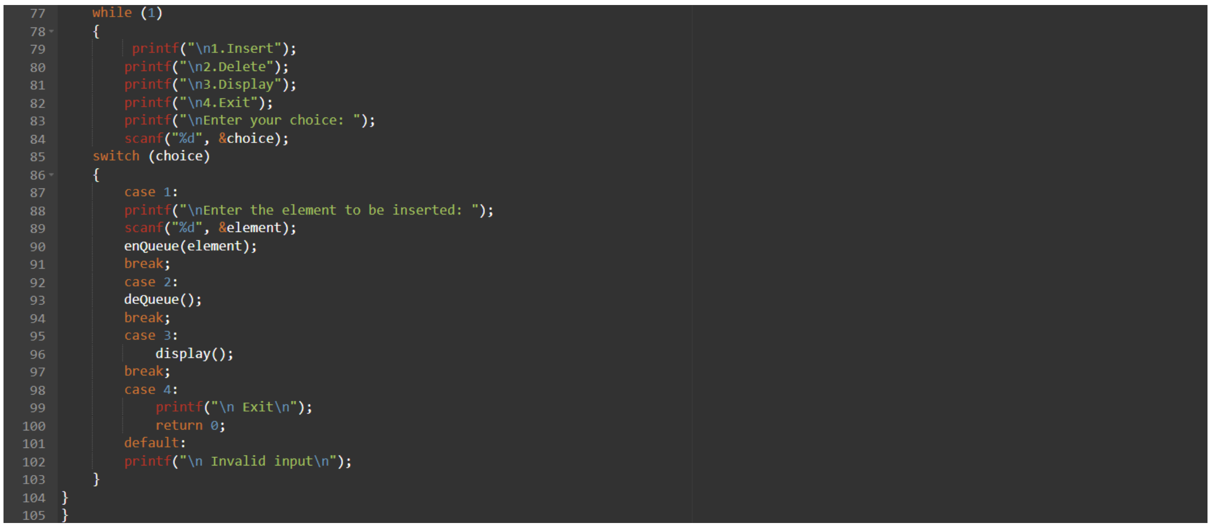
S.Praveen kumar
ch.en.u4aie22048

## Creation of Circular Queue in array:

### Program:

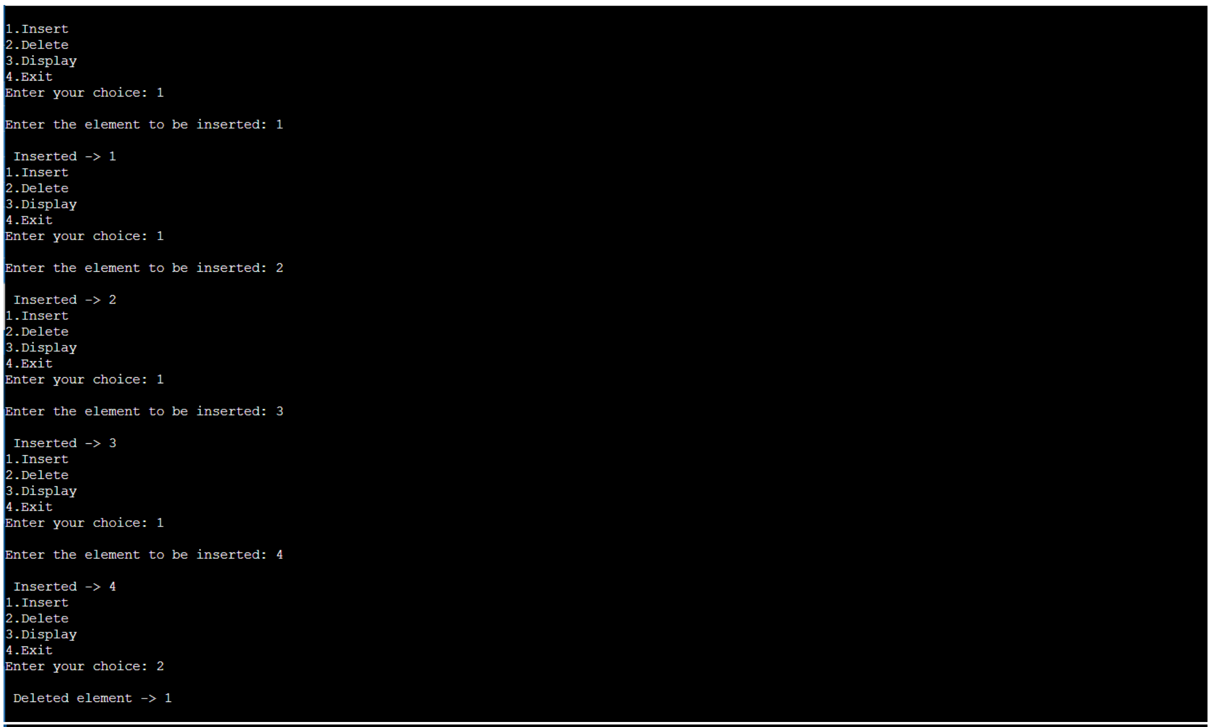

main.c

```c
//S.Praveen Kumar
//ch.en.u4aie22048
//ds-1 lab-9

//Creation of queue in linked list
#include <stdio.h>
#include <stdbool.h>
#define SIZE 5
int items[SIZE];
int front = -1, rear = -1;
bool isFull()
{
    if ((front == rear + 1) || (front == 0 && rear == SIZE - 1))
    return true;
    return false;
}
bool isEmpty()
{
    if (front == -1)
    return true;
    return false;
}
void enQueue(int element)
{
    if (isFull())
    printf("\n Queue is full\n");
    else
    {
        if (front == -1)
        front = 0;
        rear = (rear + 1) % SIZE;
        items[rear] = element;
        printf("\n Inserted -> %d", element);
    }
}
int deQueue()
{
    int element;
    if (isEmpty())
    {
        printf("\n Queue is empty\n");
        return (-1);
    }
    else
    {
        element = items[front];
    if (front == rear)
    {
        front = -1;
        rear = -1;
    }
    else
    {
        front = (front + 1) % SIZE;
    }
    printf("\n Deleted element -> %d \n", element);
    return (element);
    }
}
void display()
{
    int i;
    if (isEmpty())
    printf(" \n Empty Queue\n");
    else
    {
        for (i = front; i != rear; i = (i + 1) % SIZE)
        {
            printf("%d ->", items[i]);
        }
        printf("%d ", items[i]);
    }
}
int main()
{
    int choice, element;
```

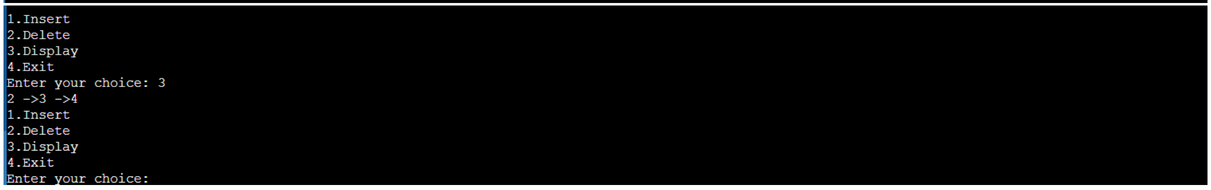


```c
    while (1)
    {
         printf("\n1.Insert");
        printf("\n2.Delete");
        printf("\n3.Display");
        printf("\n4.Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
    switch (choice)
    {
        case 1:
        printf("\nEnter the element to be inserted: ");
        scanf("%d", &element);
        enQueue(element);
        break;
        case 2:
        deQueue();
        break;
        case 3:
            display();
        break;
        case 4:
            printf("\n Exit\n");
            return 0;
        default:
        printf("\n Invalid input\n");
    }
}
}
```

## Output:



```
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1

Enter the element to be inserted: 1

 Inserted -> 1
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1

Enter the element to be inserted: 2

 Inserted -> 2
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1

Enter the element to be inserted: 3

 Inserted -> 3
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 1

Enter the element to be inserted: 4

 Inserted -> 4
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 2

 Deleted element -> 1
```



```
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice: 3
2 ->3 ->4
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
```