DATA STRUCTURES PRACTICAL EXAMINATION

Aim

Create a Doubly Linked List and perform following operations: Insertion at front and Deletion at end and at a particular position

NAME: Syeda Reeha Quasar
ROLLNO: 14114802719
GROUP: C6

Data Structures

Aim: Create a Doubly linked list and perform following operations: Insertion at front and Deletion at end

Insertion: Doubly Linked List

SOURCE CODE

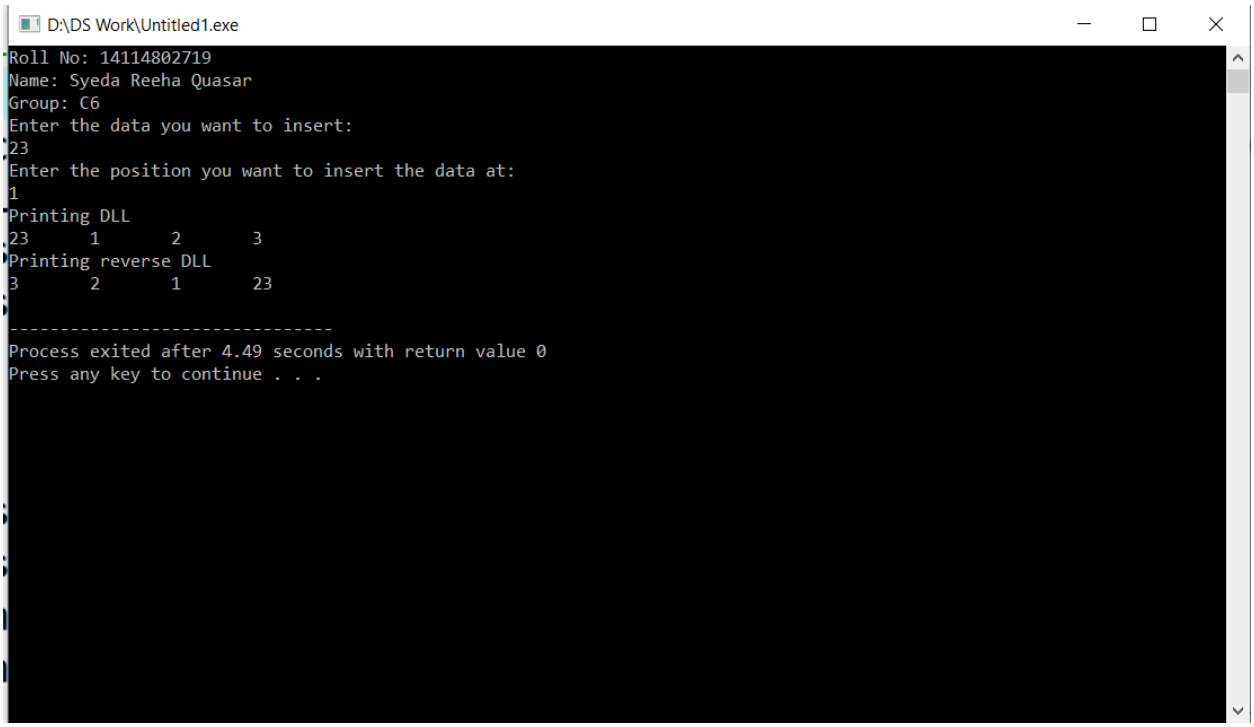
```
#include <stdio.h>

#include <stdlib.h>


struct Node {                                          //Node for Doubly Linked List
        int data;
        struct Node* previous;
        struct Node* next;
};


void insertNode (struct Node** head, int n, int m) {          //Function for Insertion
        struct Node* newNode = NULL;
        struct Node* current = *head;
        newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = n;
        if(m == 1){                                           //Insertion at beginning
                newNode->previous = NULL;
                newNode->next = *head;
                current->previous = newNode;
                *head = newNode;
        }else{                                                //Insertion at end or desired position
                int i=1;
                while (i<(m-1)){                              //Searching the element
                        current = current->next;
```

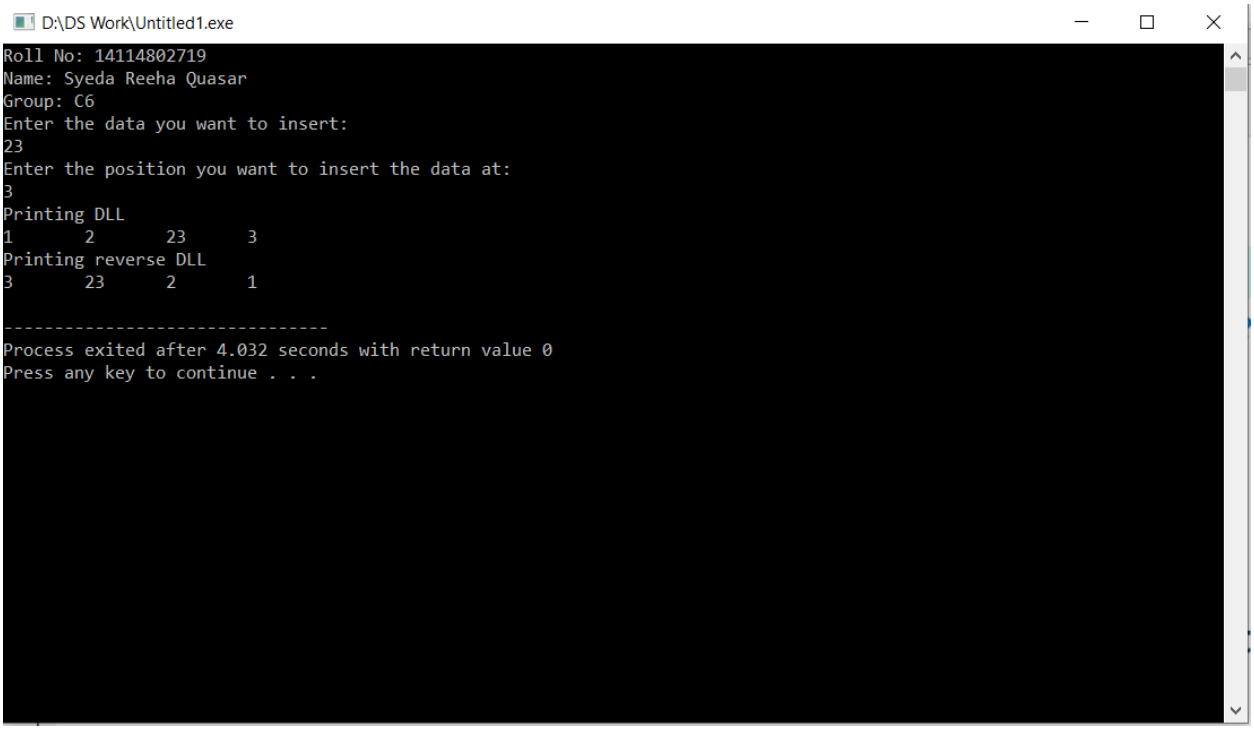
```
                i++;
            }
            if(m == 4){
                newNode->next = NULL;
            }else{
                newNode->next = current->next;
                current->next->previous = newNode;
            }
            newNode->previous = current;
            current->next = newNode;
        }
}
void printFront(struct Node* n){                    //Function for printing the Linked List using NEXT
    printf("Printing DLL\n");
    while (n != NULL) {
        printf("%d\t",n->data);
        n = n->next;
    }
    printf("\n");
}
void printBack(struct Node* n){                     //Function for printing the Linked List using PREVIOUS
    while (n->next != NULL) {
        n = n->next;
    }
    printf("Printing reverse DLL\n");
```

```
        while (n != NULL){
                printf("%d\t",n->data);
                n = n->previous;
        }
        printf("\n");
}

int main(){
        printf("Roll No: 14114802719\nName: Syeda Reeha Quasar\nGroup: C6\n");
        int n,m;
        struct Node* first =NULL;                                   //Linked List
        struct Node* second = NULL;
        struct Node* third = NULL;
        first = (struct Node*)malloc(sizeof(struct Node));
        second = (struct Node*)malloc(sizeof(struct Node));
        third = (struct Node*)malloc(sizeof(struct Node));
        first->data = 1;
        first->previous = NULL;
        first->next = second;

        second->data = 2;
        second->previous = first;
        second->next = third;

        third->data = 3;
        third->previous = second;
```

```
	third->next = NULL;


	struct Node* head = first;


	printf("Enter the data you want to insert:\n");			//Input element to be inserted
	scanf("%d", &n);
	printf("Enter the position you want to insert the data at:\n");	//Input position to insert at
	scanf("%d", &m);


	insertNode(&head,n,m);						//Function calling


	printFront(head);						//Output new Linked List from front
	printBack(head);						//Output new Linked List from back
	return 0;
}
```

OUTPUT





Source code:

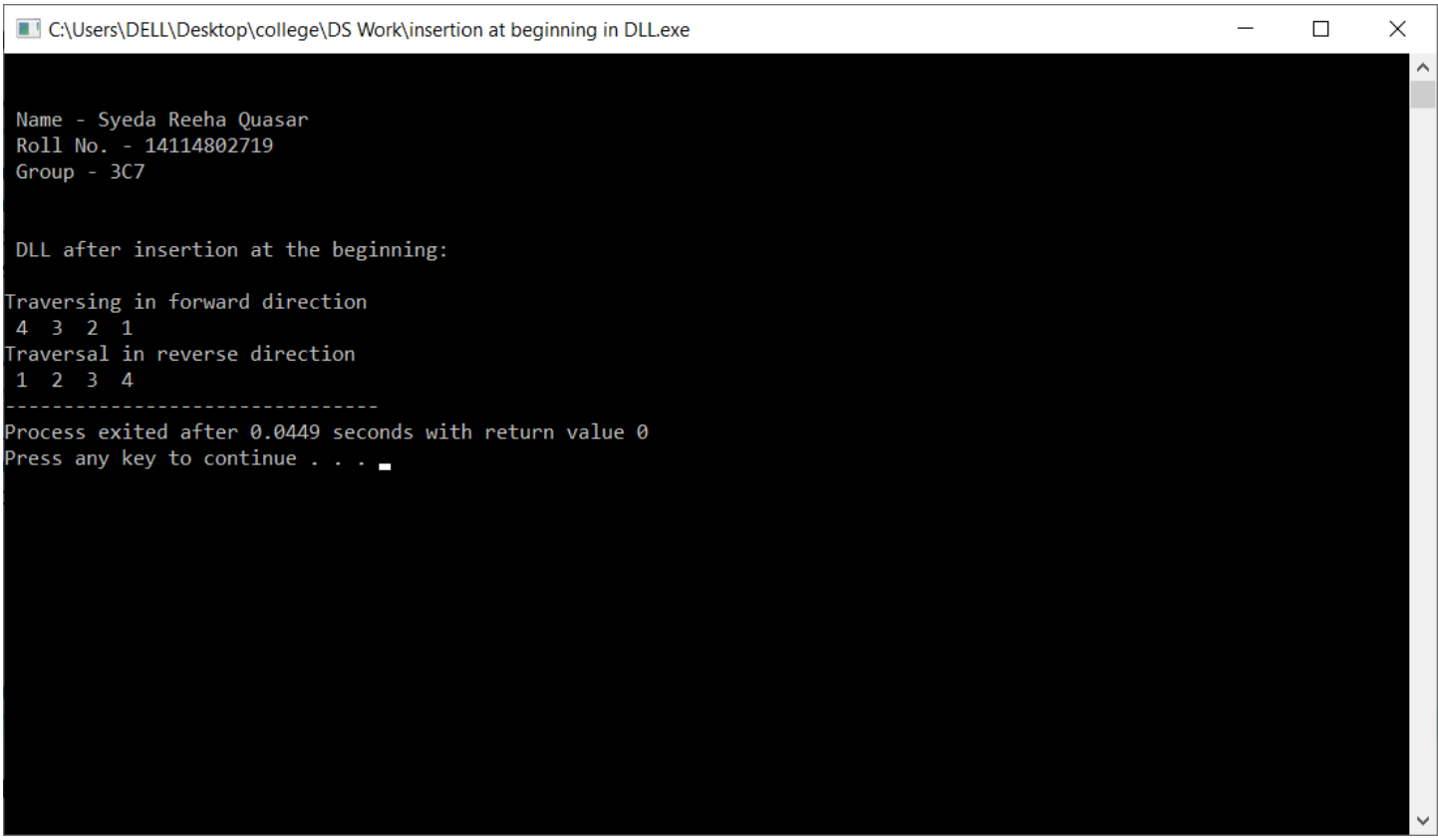
// insertion in the beginning doubly linked list

```
//required libraries
#include <stdio.h>
#include <stdlib.h>

// doubly linked list declaration
struct Node {
    int data;
    struct Node* next; // Pointer to next node
    struct Node* prev; // Pointer to previous node
};

//function for insertion at the beginning
void insertAtBeginning(struct Node** head_ref, int newHeadData)
{
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); // new node

    // assigning value to the new node and assigning values to its pointers
    new_node->data = newHeadData; //storing given data into new node
    new_node->next = (*head_ref); //dll next of node to head
    new_node->prev = NULL;  // making new node head by pointing previous node to
null

    if ((*head_ref) != NULL) // checking if head is present or not
        (*head_ref)->prev = new_node; // changing poiter of headnode from null to new
node

    (*head_ref) = new_node; // changing head to new node
}
```

```
// printing the DLL
void printList(struct Node* node)
{
    struct Node* last;  // declaring a new node for reverse traversal

    // traversal in forward direction
    printf("\nTraversing in forward direction \n");
    while (node != NULL) {
        printf(" %d ", node->data);
        last = node;
        node = node->next;
    }

        //traversal in reverse direction
    printf("\nTraversal in reverse direction \n");
    while (last != NULL) {
        printf(" %d ", last->data);
        last = last->prev;
    }
}

int main()
{
        // my info
        printf("\n\n Name - Syeda Reeha Quasar \n Roll No. - 14114802719 \n Group - 3C7 \n\n");

        struct Node* head = NULL; // declaring head as null
```

```
    // all these elements are inserted in the beginning
    insertAtBeginning(&head, 1); //inserting 1
    insertAtBeginning(&head, 2); //inserting 2
    insertAtBeginning(&head, 3); //inserting 3
    insertAtBeginning(&head, 4); //inserting 4

    printf("\n DLL after insertion at the beginning: \n");
    printList(head); // printing DLL

    return 0;
}
```

OUTPUT



Deletion: Doubly Linked List

SOURCE CODE

```
// deletion from the end doubly linked list

//required libraries
#include <stdio.h>
#include <stdlib.h>

// doubly linked list declaration
struct Node {
    int data;
    struct Node* next; // Pointer to next node
    struct Node* prev; // Pointer to previous node
};

//function for insertion at the beginning
void insertAtBeginning(struct Node** head_ref, int newHeadData)
{
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); // new node

    // assigning value to the new node and assigning values to its pointers
    new_node->data = newHeadData; //storing given data into new node
    new_node->next = (*head_ref); //dll next of node to head
    new_node->prev = NULL;  // making new node head by pointing previous node to
null
```

```
    if ((*head_ref) != NULL) // checking if head is present or not
        (*head_ref)->prev = new_node; // changing poiter of headnode from null to new
node


    (*head_ref) = new_node; // changing head to new node
}

void deletionAtEnd(struct Node* node)
{
          //traversing the list to find second lastnode
    while (node -> next -> next != NULL) {
        node = node->next;
    }

    node -> next = NULL; // changing second last node pointer to null
}

// printing the DLL
void printList(struct Node* node)
{
    struct Node* last;  // declaring a new node for reverse traversal

    // traversal in forward direction
    printf("\nTraversing in forward direction \n");
    while (node != NULL) {
        printf(" %d ", node->data);
        last = node;
        node = node->next;
```

```
    }

        //traversal in reverse direction
    printf("\nTraversal in reverse direction \n");
    while (last != NULL) {
        printf(" %d ", last->data);
        last = last->prev;
    }
}

int main()
{
        // my info
        printf("\n\n Name - Syeda Reeha Quasar \n Roll No. - 14114802719 \n Group -
3C7 \n\n");

        struct Node* head = NULL; // declaring head as null

        // all these elements are inserted in the beginning
        insertAtBeginning(&head, 1); //inserting 1
        insertAtBeginning(&head, 2); //inserting 2
        insertAtBeginning(&head, 3); //inserting 3
        insertAtBeginning(&head, 4); //inserting 4

        printf("\n DLL before dletion: \n");
        printList(head); // printing DLL

        printf("\n\n\n\n DLL after deletion from the end: \n");
        deletionAtEnd(head);
```

```
        printList(head); // printing DLL


        return 0;
}
```

OUTPUT