ATMA RAM SANATAN DHARMA COLLEGE

UNIVERSITY OF DELHI

DSC 07: DATA STRUCTURES

SEM III

ASSIGNEMENT 12

Submitted by:
Shishirant Singh
22/28081
B.Sc. Hons. Computer Science

Submitted to:
Ms. Archana Gahlaut

DOUBLY LINKED LIST

CODE:

```
// C program for the all operations in
// the Doubly Linked List
#include <stdio.h>
#include <stdlib.h>

// Linked List Node
struct node {
	int info;
	struct node *prev, *next;
};
struct node* start = NULL;

// Function to traverse the linked list
void traverse()
{
	// List is empty
	if (start == NULL) {
		printf("\nList is empty\n");
		return;
	}
	// Else print the Data
	struct node* temp;
	temp = start;
	while (temp != NULL) {
		printf("Data = %d \n", temp->info);
		temp = temp->next;
	}
}

// Function to insert at the front
// of the linked list
void insertAtFront()
{
	int data;
	struct node* temp;
	temp = (struct node*)malloc(sizeof(struct node));
```

```
        printf("\nEnter number to be inserted: ");
        scanf("%d", &data);
        temp->info = data;
        temp->prev = NULL;

        // Pointer of temp will be
        // assigned to start
        temp->next = start;
        start = temp;
}

// Function to insert at the end of
// the linked list
void insertAtEnd()
{
        int data;
        struct node *temp, *trav;
        temp = (struct node*)malloc(sizeof(struct node));
        temp->prev = NULL;
        temp->next = NULL;
        printf("\nEnter number to be inserted: ");
        scanf("%d", &data);
        temp->info = data;
        temp->next = NULL;
        trav = start;

        // If start is NULL
        if (start == NULL) {

                start = temp;
        }

        // Changes Links
        else {
                while (trav->next != NULL)
                        trav = trav->next;
                temp->prev = trav;
                trav->next = temp;
        }
}
```

```
// Function to insert at any specified
// position in the linked list
void insertAtPosition()
{
	int data, pos, i = 1;
	struct node *temp, *newnode;
	newnode = malloc(sizeof(struct node));
	newnode->next = NULL;
	newnode->prev = NULL;

	// Enter the position and data
	printf("\nEnter position : ");
	scanf("%d", &pos);


	// If start==NULL,
	if (start == NULL) {
		start = newnode;
		newnode->prev = NULL;
		newnode->next = NULL;
	}

	// If position==1,
	else if (pos == 1) {
	// this is author method its correct but we can simply call insertAtfront()
function for this special case
	/* newnode->next = start;
		newnode->next->prev = newnode;
		newnode->prev = NULL;
		start = newnode; */
	// now this is improved by Jay Ghughriwala on geeksforgeeks
	insertAtFront();
	}

	// Change links
	else {
	printf("\nEnter number to be inserted: ");
	scanf("%d", &data);
	newnode->info = data;
```

```
        temp = start;
                while (i < pos - 1) {
                        temp = temp->next;
                        i++;
                }
                newnode->next = temp->next;
                newnode->prev = temp;
                temp->next = newnode;
                temp->next->prev = newnode;
        }
}

// Function to delete from the front
// of the linked list
void deleteFirst()
{
        struct node* temp;
        if (start == NULL)
                printf("\nList is empty\n");
        else {
                temp = start;
                start = start->next;
                if (start != NULL)
                        start->prev = NULL;
                free(temp);
        }
}

// Function to delete from the end
// of the linked list
void deleteEnd()
{
        struct node* temp;
        if (start == NULL)
                printf("\nList is empty\n");
        temp = start;
        while (temp->next != NULL)
                temp = temp->next;
        if (start->next == NULL)
                start = NULL;
```

```
        else {
                temp->prev->next = NULL;
                free(temp);
        }
}

// Function to delete from any specified
// position from the linked list
void deletePosition()
{
        int pos, i = 1;
        struct node *temp, *position;
        temp = start;

        // If DLL is empty
        if (start == NULL)
                printf("\nList is empty\n");

        // Otherwise
        else {
                // Position to be deleted
                printf("\nEnter position : ");
                scanf("%d", &pos);

                // If the position is the first node
                if (pos == 1) {
                        deleteFirst(); // im,proved by Jay Ghughriwala on
GeeksforGeeks
                        if (start != NULL) {
                                start->prev = NULL;
                        }
                        free(position);
                        return;
                }

                // Traverse till position
                while (i < pos - 1) {
                        temp = temp->next;
                        i++;
                }
```

```
                // Change Links
                position = temp->next;
                if (position->next != NULL)
                        position->next->prev = temp;
                temp->next = position->next;

                // Free memory
                free(position);
        }
}

// Driver Code
int main()
{
        int choice;
        while (1) {

                printf("\n\t1- To see list\n");
                printf("2- For insertion at"
                        " starting\n");
                printf("3- For insertion at"
                        " end\n");
                printf("4- For insertion at "
                        "any position\n");
                printf("5- For deletion of "
                        "first element\n");
                printf("6- For deletion of "
                        "last element\n");
                printf("7- For deletion of "
                        "element at any position\n");
                printf("8- To exit\n");
                printf("\nEnter Choice :\n");
                scanf("%d", &choice);

                switch (choice) {
                case 1:
                        traverse();
                        break;
                case 2:
                        insertAtFront();
```

```
                break;
            case 3:
                insertAtEnd();
                break;
            case 4:
                insertAtPosition();
                break;
            case 5:
                deleteFirst();
                break;
            case 6:
                deleteEnd();
                break;
            case 7:
                deletePosition();
                break;

            case 8:
                exit(1);
                break;
            default:
                printf("Incorrect Choice. Try Again \n");
                continue;
            }
        }
        return 0;
}
```

OUTPUT:

```
Output
Enter Choice :
2
Enter number to be inserted: 7
1- To see list
2- For insertion at starting
3- For insertion at end
4- For insertion at any position
5- For deletion of first element
6- For deletion of last element
7- For deletion of element at any position
8- To exit

Enter Choice :
1
Data = 7
Data = 5
Data = 4

    1- To see list
2- For insertion at starting
3- For insertion at end
4- For insertion at any position
5- For deletion of first element
6- For deletion of last element
7- For deletion of element at any position
```

Output

```
    1- To see list
2- For insertion at starting
3- For insertion at end
4- For insertion at any position
5- For deletion of first element
6- For deletion of last element
7- For deletion of element at any position
8- To exit

Enter Choice :
6
1- To see list
2- For insertion at starting
3- For insertion at end
4- For insertion at any position
5- For deletion of first element
6- For deletion of last element
7- For deletion of element at any position
8- To exit

Enter Choice :
1
Data = 7
Data = 5
```