Code

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

/* QUESTION:
Create a singly linked list to store name, roll no and marks obtained by n students. Implement
the following functions:
    i) Insert student at front of list
    ii) Insert student at end of list
    iii) Insert student after a particular student (roll no)
    iv) Delete student at front of list
    v) Delete student at end of list
    vi) Delete a student having a particular roll no.
*/

#define ARR_LEN 256

// Structure to hold Linked List Node
struct Node{
    char name[ARR_LEN];
    int roll;
    float marks;
    struct Node *next;
};
// Initialize a HEAD to blank linked list.
struct Node *head = NULL;

// Function to input node data from user and return it's address
struct Node *makeNode(){
    struct Node *newNode = malloc(sizeof(struct Node));
    printf("Enter name: ");
    scanf("%s", newNode->name);
    printf("Enter roll: ");
    scanf("%d", &newNode->roll);
    printf("Enter marks: ");
    scanf("%f", &newNode->marks);
    return newNode;
}

// Adds node to front of LL
void addFront(){
    struct Node *newNode = makeNode();
    newNode->next = (head == NULL) ? NULL : head;
    head = newNode;
}

// Removes node at front of LL
void removeFront(){
    // Do nothing if blank list
    if (head == NULL)
        return;
    struct Node *temp = head->next;
    free(head);
    head = temp;
}
```

```
// Adds node to end of Linked list
void addEnd(){
    struct Node *newNode = makeNode(), *end;
    // If blank, set newNode to HEAD of list.
    if (head == NULL)
        head = newNode;
    else
    {
        for (end = head; end->next != NULL; end = end->next);
        end->next = newNode;
    }
    newNode->next = NULL;
}

// Removes a node from the end of list
void removeEnd(){
    struct Node *end, *secEnd;
    // Do nothing if empty list.
    if (head == NULL)
        return;
    for (end = head; end->next != NULL; end = end->next)
        secEnd = end;
    // if second to end node is NULL, only one node in list
    if (secEnd == NULL)
        head = NULL;
    else
        secEnd->next = NULL;
    free(end);
}

// Adds a node after node specified by roll number
void addAfterRoll(int rk){
    struct Node *newNode = makeNode(), *pos;
    for (pos = head; pos->roll != rk && pos != NULL; pos = pos->next)
        ;
    // Do nothing if not found or blank list.
    if (pos == NULL)
        return;
    newNode->next = pos->next;
    pos->next = newNode;
}

// Deletes a node specified by roll number
void deleteRoll(int rk){
    struct Node *pos, *secPos = NULL;
    for (pos = head; pos->roll != rk && pos != NULL; pos = pos->next)
        secPos = pos;
    // Do nothing if not found or blank list.
    if (pos == NULL)
        return;
    // second to end node is NULL, only one node in list
    if (secPos == NULL)
        head = pos->next;
    else
        secPos->next = pos->next;
    free(pos);
}
```

```c
// Shows whole list as table
void showList(){
    printf("Roll\tName\tMarks\n");
    for (struct Node *n = head; n != NULL; n = n->next)
        printf("%d\t%s\t%.2f\n", n->roll, n->name, n->marks);
    printf("\n");
}

// Driver code for Menu UI
void main(){
    int t;
    printf("Student Roll\n------------\n(A) Insert Front\t(B) Delete Front\n(C) Insert End\t\t(D) Delete End\n(E) Add After Roll\t(F) Delete Roll\n(G) Show List\t\t(H) EXIT\n");
    while(1){
        printf("Option: ");
        fflush(stdin);
        char c = getc(stdin);
        switch(c){
            case 'A':
                addFront();
                break;
            case 'B':
                removeFront();
                break;
            case 'C':
                addEnd();
                break;
            case 'D':
                removeEnd();
                break;
            case 'E':
                printf("Enter Roll: ");
                scanf("%d", &t);
                addAfterRoll(t);
                break;
            case 'F':
                printf("Enter Roll: ");
                scanf("%d", &t);
                deleteRoll(t);
                break;
            case 'G':
                showList();
                break;
            case 'H':
                return;
            default:
                printf("Invalid Option!!\n");
        }
    }
}
```

Output

```
PS Z:\DSLab> gcc .\studentData.c; .\a.exe
Student Roll
------------
(A) Insert Front        (B) Delete Front
(C) Insert End          (D) Delete End
(E) Add After Roll      (F) Delete Roll
(G) Show List           (H) EXIT
Option: A
Enter name: Anil
Enter roll: 10
Enter marks: 44
Option: C
Enter name: Binu
Enter roll: 20
Enter marks: 55
Option: G
Roll    Name    Marks
10      Anil    44.00
20      Binu    55.00

Option: E
Enter Roll: 10
Enter name: Cathy
Enter roll: 30
Enter marks: 66
Option: G
Roll    Name    Marks
10      Anil    44.00
30      Cathy   66.00
20      Binu    55.00

Option: F
Enter Roll: 20
Option: G
Roll    Name    Marks
10      Anil    44.00
30      Cathy   66.00

Option: B
Option: G
Roll    Name    Marks
30      Cathy   66.00

Option: A
Enter name: Daniel
Enter roll: 40
Enter marks: 77
Option: G
Roll    Name    Marks
40      Daniel  77.00
30      Cathy   66.00

Option: D
Option: G
Roll    Name    Marks
40      Daniel  77.00

Option: H
PS Z:\DSLab>
```