Folder SEM 3\Exp1

1 printable files

(file list disabled)

SEM 3\Exp1\SLL_implementation.c

```c
// program to implement singly linked list

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

// Core structure of Node that forms Linked List
struct node
{
    int data;
    struct node *next;
};

// End of the linked list should not point to anything(NULL)
struct node *head = NULL;

// Function to insert a node at the beginning of the list
void insertFirst(int data)
{
    struct node *new_node = (struct node *)malloc(sizeof(struct node));

    new_node→data = data;
    new_node→next = head;

    head = new_node;
}

// Function to insert a node at the end of the list
void insertEnd(int data)
{
    struct node *new_node = (struct node *)malloc(sizeof(struct node));

    new_node→data = data;
    new_node→next = NULL;

    if (head == NULL)
    {
        head = new_node;
        return;
    }

    struct node *temp = head;

    while (temp→next ≠ NULL)
    {
        temp = temp→next;
```

```
46          }
47
48          temp→next = new_node;
49  }
50
51  void insertPos(int data, int pos)
52  {
53          struct node *new_node = (struct node *)malloc(sizeof(struct node));
54
55          new_node→data = data;
56
57          int curr_pos = 0;
58          struct node *temp = head;
59
60          while (temp→next ≠ NULL && curr_pos < pos - 1)
61          {
62              temp = temp→next;
63              curr_pos++;
64          }
65
66          new_node→next = temp→next;
67          temp→next = new_node;
68  }
69
70  void deleteFirst()
71  {
72          if (head == NULL)
73          {
74              printf("List is empty");
75              return;
76          }
77
78          struct node *temp = head;
79          head = head→next;
80          free(temp);
81  }
82
83  void deleteEnd()
84  {
85          if (head == NULL)
86          {
87              printf("List is empty");
88              return;
89          }
90
91          struct node *temp = head;
92          struct node *prev = NULL;
93
94          while (temp→next ≠ NULL)
95          {
96              prev = temp;
97              temp = temp→next;
98          }
99
```

```c
100        prev→next = NULL;
101        free(temp);
102    }
103
104    void deletePos(int pos)
105    {
106        if (head == NULL)
107        {
108            printf("List is empty");
109            return;
110        }
111
112        struct node *temp = head;
113        struct node *prev = NULL;
114        int curr_pos = 0;
115
116        while (temp→next != NULL && curr_pos < pos - 1)
117        {
118            prev = temp;
119            temp = temp→next;
120            curr_pos++;
121        }
122
123        prev→next = temp→next;
124        free(temp);
125    }
126
127    void display()
128    {
129        struct node *temp = head;
130
131        while (temp != NULL)
132        {
133            printf("%d → ", temp→data);
134            temp = temp→next;
135        }
136        printf("NULL\n");
137    }
138
139    int main()
140    {
141        printf("Linked List creation and Manipulation\n");
142        printf("Enter from the following options:\n");
143        printf("1. Insert at the beginning of the list\n");
144        printf("2. Insert at the end of the list\n");
145        printf("3. Insert at a specific position in the list\n");
146        printf("4. Delete from the beginning of the list\n");
147        printf("5. Delete from the end of the list\n");
148        printf("6. Delete from a specific position in the list\n");
149        printf("7. Display the list\n");
150        printf("8. Exit\n");
151
152        int choice;
153        int data;
```

```c
154        int pos;
155
156        while (1)
157        {
158            printf("Enter your choice: ");
159            scanf("%d", &choice);
160
161            switch (choice)
162            {
163            case 1:
164                printf("Enter the data to be inserted: ");
165                scanf("%d", &data);
166                insertFirst(data);
167                break;
168            case 2:
169                printf("Enter the data to be inserted: ");
170                scanf("%d", &data);
171                insertEnd(data);
172                break;
173            case 3:
174                printf("Enter the data to be inserted: ");
175                scanf("%d", &data);
176                printf("Enter the position to insert the data: ");
177                scanf("%d", &pos);
178                insertPos(data, pos);
179                break;
180            case 4:
181                deleteFirst();
182                break;
183            case 5:
184                deleteEnd();
185                break;
186            case 6:
187                printf("Enter the position to delete the data: ");
188                scanf("%d", &pos);
189                deletePos(pos);
190                break;
191            case 7:
192                display();
193                break;
194            case 8:
195                exit(0);
196            default:
197                printf("Invalid choice");
198                break;
199            }
200        }
201
202        return 0;
203    }
```