

MIT WORLD PEACE UNIVERSITY

Object Oriented Programming with Java and C++  
Second Year B. Tech, Semester 1

---

---

SINGLE LINKED LIST OPERATIONS

---

---

PRACTICAL REPORT  
ASSIGNMENT 5

Prepared By

Krishnaraj Thadesar  
Cyber Security and Forensics  
Batch A2, PA 20

November 13, 2022

# Contents

<b>1 Objectives</b>	<b>1</b>
<b>2 Problem Statements</b>	<b>1</b>
<b>3 Theory</b>	<b>1</b>
3.1 Singly Linked Lists . . . . .	1
3.1.1 Purpose of Head Node in Singly Linked List . . . . .	2
3.2 Various Operations on a Singly Linked List . . . . .	2
<b>4 Platform</b>	<b>2</b>
<b>5 Input</b>	<b>2</b>
<b>6 Output</b>	<b>3</b>
<b>7 Test Conditions</b>	<b>3</b>
<b>8 Code</b>	<b>3</b>
8.1 Pseudo Code . . . . .	3
8.1.1 Pseudo Code for Creatoin of a Singly Linked List . . . . .	3
8.1.2 Pseudo Code for Display of a Singly Linked List . . . . .	3
8.1.3 Pseudo Code for Insertion in a Singly Linked List . . . . .	3
8.1.4 Pseudo Code for Deletion in a Singly Linked List . . . . .	4
8.1.5 Pseudo Code for Reversing of a Singly Linked List . . . . .	4
8.1.6 Pseudo Code for Sorting of a Singly Linked List . . . . .	4
8.1.7 Pseudo Code for Merging of 2 Singly Linked Lists . . . . .	5
8.2 C Implementation of Problem Statement . . . . .	5
8.3 Input and Output . . . . .	11
<b>9 Time Complexity</b>	<b>19</b>
<b>10 Conclusion</b>	<b>20</b>
<b>11 FAQs</b>	<b>20</b>

### 1 Objectives

1. To study data structure: Singly Linked List
2. To Study different operations that could be performed on SLL.
3. To Study Applications of Singly Linked list

### 2 Problem Statements

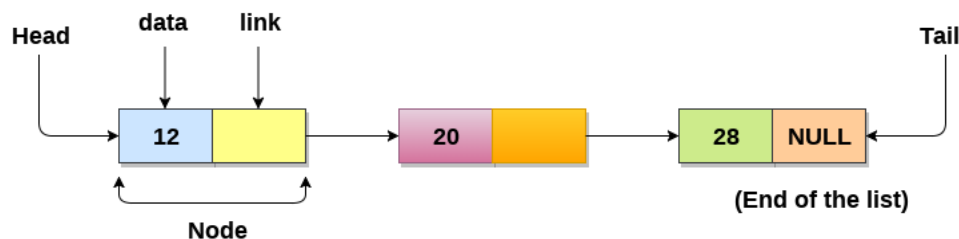
*Department of Computer Engineering has student's node named 'Pinnacle Node'. Students of second, third and final year of department can be granted membership on request. Similarly, one may cancel the membership of node. First node is reserved for president of node and last node is reserved for the secretary of the node. Write C program to maintain node members information using singly linked list. Store student PRN and Name. Write functions to:*

1. Add members as well as president or even secretary.
2. Compute total number of members of node
3. Display members
4. sorting of two linked list
5. merging of two linked list
6. Reversing using three pointers
7. Add and delete the

### 3 Theory

#### 3.1 Singly Linked Lists

Linked List can be defined as collection of objects called nodes that are randomly stored in the memory. A node contains two fields i.e. data stored at that particular address and the pointer which contains the address of the next node in the memory. The last node of the list contains pointer to the null.



### **3.1.1 Purpose of Head Node in Singly Linked List**

The Head Node in a Single Linked List is the most important node.

- It points to the second node, which then points to the next node, and so on, essentially pointing to the entire linked list, which is scattered randomly in the memory.
- Losing the head node would mean losing the entire linked list.
- The head node is just a single node, and passing it around functions makes it space efficient as opposed to passing the entire linked list or an entire array.
- The Head node being used just as a pointer to point to the entire linked list ensures that any function can perform any operation on the linked list by just taking the head node as a parameter.

### **3.2 Various Operations on a Singly Linked List**

A *Singly Linked List* being an Abstract Data Type, has a variety of operations that you can perform on it, like:

1. Searching: Look for a particular element with some key data in the entire linked list by traversing through it.
2. Sorting: Sorting elements of a linked list just like any other data structure.
3. Deleting: Deleting any element of the linked list by traversing through it.
4. Inserting: Inserting an element at a given position in the linked list.
5. Merging: Merging 2 Linked lists in any way desirable, be it by merging their tails to the heads, or by creating copies of both the linked lists, and then returning the head of the copy.
6. Displaying, etc.

The Pseudo Codes for these functions are written further in the paper.

## **4 Platform**

**Operating System:** Arch Linux x86-64

**IDEs or Text Editors Used:** Visual Studio Code

**Compilers :** gcc on linux for C

## **5 Input**

- Atleast 5 Elements to Input, including the President and the Secretary
- Details of Every Element like Name and PRN
- Options to Select what to do.

## 6 Output

- Menu to display all the operations you can perform on the Linked list.
- Display of All the elements of the Linked list, before and after performing operations on it.

## 7 Test Conditions

1. Input at least 5 records.
2. Inserting an Element at All Positions
3. Delete an Element from All positions

## 8 Code

### 8.1 Pseudo Code

#### 8.1.1 Pseudo Code for Creatoin of a Singly Linked List

```
1  struct node *head, *merged_head;
2  head = (struct node *)malloc(sizeof(struct node));
3  head->next = NULL;
4  // then add new nodes.
5  struct node *temp = head;
6  while(enter elements)
7      struct node *curr = (struct node *)malloc(sizeof(struct node));
8      scanf("%s", curr->Data);
9      curr->next = NULL;
10     temp->next = curr;
11     temp = curr;
```

#### 8.1.2 Pseudo Code for Display of a Singly Linked List

```
1  if (head->next == NULL)
2      print("\nNo members");
3      return -1;
4  struct node *curr = (struct node *)malloc(sizeof(struct node));
5  curr = head->next;
6  while (curr != NULL)
7      print("\n--Member: %d--", ++counter);
8      print("\Data: %s", curr->data);
9      curr = curr->next;
10  printf("\n");
```

#### 8.1.3 Pseudo Code for Insertion in a Singly Linked List

```
1  struct node *curr = head;
2  struct node *nnode = (struct node *)malloc(sizeof(struct node));
3  print(Enter data)
4  scan(pos)
5  pos--;
6  int total_length = findLength(head);
```

```
7     if (pos > total_length + 1)
8         printf("Data cant be inserted!\n");
9         return 0;
10    else
11        while (curr != NULL && i < total_length)
12            i++;
13            curr = curr->next;
14            nnode->next = curr->next;
15            curr->next = nnode;
```

### 8.1.4 Pseudo Code for Deletion in a Singly Linked List

```
1     if (head->next == NULL)
2         print("list empty")
3         return -1;
4         struct node *prev = (struct node *)malloc(sizeof(struct node *));
5         prev = head;
6         int count = 1;
7         struct node *curr = (struct node *)malloc(sizeof(struct node *));
8         curr = head->next;
9
10        int len = find_total_members(head);
11
12        if (position > len)
13            scanf("\nCant Delete data");
14        else
15            while (count < position && curr->next != NULL)
16                count++;
17                prev = curr;
18                curr = curr->next;
19            struct node *temp = (struct node *)malloc(sizeof(struct node *));
20
21            temp = curr;
22            prev->next = curr->next;
23            curr->next = NULL;
24            free(temp);
```

### 8.1.5 Pseudo Code for Reversing of a Singly Linked List

```
1     struct node *current = head->next;
2     struct node *prev = NULL, *future = NULL;
3     while (current != NULL)
4         future = current->next;
5         current->next = prev;
6         prev = current;
7         current = future;
8     head->next = prev;
```

### 8.1.6 Pseudo Code for Sorting of a Singly Linked List

```
1     struct node *i = (struct node *)malloc(sizeof(struct node));
2     struct node *j = (struct node *)malloc(sizeof(struct node));
3     struct node temp;
4
5     // bubble sorting
```

```
6     for (i = head; i != NULL; i = i->next)
7         for (j = i; j->next != NULL; j = j->next)
8             if (j->prn > j->next->prn)
9                 temp.prn = j->prn;
10                j->prn = j->next->prn;
11                j->next->prn = temp.prn;
```

### 8.1.7 Pseudo Code for Merging of 2 Singly Linked Lists

```
1     struct node *merged_head = (struct node *)malloc(sizeof(struct node));
2     merged_head->next = NULL;
3     if (head->next == NULL || head_2->next == NULL)
4         printf("\nOne of the nodes is empty, so no point in merging!\n");
5     return merged_head;
6
7     struct node *temp_head, *temp_merged, *current;
8     temp_head = head->next;
9
10    temp_merged = (struct node *)malloc(sizeof(struct node));
11    merged_head->next = temp_merged;
12    while (temp_head != NULL)
13        current = temp_merged;
14        temp_merged->prn = temp_head->prn;
15
16        temp_merged = (struct node *)malloc(sizeof(struct node));
17        current->next = temp_merged;
18        temp_head = temp_head->next;
19
20    temp_head = head_2->next;
21    while (temp_head != NULL)
22        current = temp_merged;
23        temp_merged->prn = temp_head->prn;
24
25        temp_merged = (struct node *)malloc(sizeof(struct node));
26        current->next = temp_merged;
27        temp_head = temp_head->next;
28
29    current->next = NULL;
30
31    return merged_head;
```

## 8.2 C Implementation of Problem Statement

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 // something about trhe president and the club.
5 // First and last are reserved for the president and the secretary.
6 // people can enroll in the club as well.
7
8 struct club
9 {
10     int prn;
11     char name[30];
12     struct club *next;
13 };
```

```
14
15 void add_member(struct club *head)
16 {
17     int choice = 1;
18     struct club *temp = head;
19
20     do
21     {
22         if (head->next == NULL)
23         {
24             printf("\nEnter Name of the President: ");
25         }
26         else
27         {
28             printf("\nEnter Name: ");
29         }
30         struct club *curr = (struct club *)malloc(sizeof(struct club));
31         scanf("%s", curr->name);
32         printf("\nEnter the PRN: ");
33         scanf("%d", &curr->prn);
34         curr->next = NULL;
35         temp->next = curr;
36         temp = curr;
37         printf("\nDo you want to enter more Members? (1 or 0)");
38         scanf("%d", &choice);
39     } while (choice != 0);
40
41     struct club *curr = (struct club *)malloc(sizeof(struct club));
42     printf("\nEnter The name of the Secretary: ");
43     scanf("%s", curr->name);
44     printf("\nEnter the PRN: ");
45     scanf("%d", &curr->prn);
46     curr->next = NULL;
47     temp->next = curr;
48     temp = curr;
49 }
50
51 int find_total_members(struct club *head)
52 {
53     struct club *curr = (struct club *)malloc(sizeof(struct club *));
54     int count = 0;
55     curr = head->next;
56     while (curr != NULL)
57     {
58         count++;
59         curr = curr->next;
60     }
61     return count;
62 }
63
64 int delete_member(struct club *head, int position)
65 {
66     if (head->next == NULL)
67     {
68         printf("\n\nList is Empty\n\n");
69         return -1;
70     }
71     struct club *prev = (struct club *)malloc(sizeof(struct club *));
72     prev = head;
```



```
73     int count = 1;
74     struct club *curr = (struct club *)malloc(sizeof(struct club *));
75
76     curr = head->next;
77
78     int len = find_total_members(head);
79     if (position > len)
80     {
81         scanf("\nCant Delete data");
82     }
83     else
84     {
85         while (count < position && curr->next != NULL)
86         {
87             count++;
88             prev = curr;
89             curr = curr->next;
90         }
91     }
92     struct club *temp = (struct club *)malloc(sizeof(struct club *));
93
94     temp = curr;
95     prev->next = curr->next;
96     curr->next = NULL;
97     free(temp);
98 }
99
100 int display_club(struct club *head)
101 {
102     int counter = 0;
103     if (head->next == NULL)
104     {
105         printf("\nNo members in the club");
106         return -1;
107     }
108     struct club *curr = (struct club *)malloc(sizeof(struct club));
109     curr = head->next;
110     while (curr != NULL)
111     {
112         printf("\n--Member: %d--", ++counter);
113         printf("\nName: %s", curr->name);
114         printf("\nPRN: %d", curr->prn);
115         curr = curr->next;
116     }
117     printf("\n");
118 }
119
120 void sort_linked_list(struct club *head)
121 {
122     struct club *i = (struct club *)malloc(sizeof(struct club));
123     struct club *j = (struct club *)malloc(sizeof(struct club));
124     struct club temp;
125
126     // bubble sorting
127     for (i = head; i != NULL; i = i->next)
128     {
129         for (j = i; j->next != NULL; j = j->next)
130         {
131             if (j->prn > j->next->prn)
```

```
132     {
133         // swapping elements of a linked list here, lengthy but with the
134         // conventional logic.
135         // we dont touch the next variable, coz that would change the
136         // order of the addresses and mess up the linked list.
137         // so we just swap the actual values.
138         temp.prn = j->prn;
139         strcpy(temp.name, j->name);
140
141         j->prn = j->next->prn;
142         strcpy(j->name, j->next->name);
143
144         j->next->prn = temp.prn;
145         strcpy(j->next->name, temp.name);
146     }
147 }
148
149 struct club *merge_2_linked_list(struct club *head, struct club *head_2)
150 {
151     struct club *merged_head = (struct club *)malloc(sizeof(struct club));
152     merged_head->next = NULL;
153     if (head->next == NULL || head_2->next == NULL)
154     {
155         printf("\nOne of the clubs is empty, so no point in merging!\n");
156         return merged_head;
157     }
158
159     struct club *temp_head, *temp_merged, *current;
160     temp_head = head->next;
161
162     temp_merged = (struct club *)malloc(sizeof(struct club));
163     merged_head->next = temp_merged;
164     while (temp_head != NULL)
165     {
166         current = temp_merged;
167         strcpy(temp_merged->name, temp_head->name);
168         temp_merged->prn = temp_head->prn;
169
170         temp_merged = (struct club *)malloc(sizeof(struct club));
171         current->next = temp_merged;
172         temp_head = temp_head->next;
173     }
174
175     temp_head = head_2->next;
176     while (temp_head != NULL)
177     {
178         current = temp_merged;
179         strcpy(temp_merged->name, temp_head->name);
180         temp_merged->prn = temp_head->prn;
181
182         temp_merged = (struct club *)malloc(sizeof(struct club));
183         current->next = temp_merged;
184         temp_head = temp_head->next;
185     }
186
187     current->next = NULL;
188 }
```

```
189     return merged_head;
190 }
191
192 void reverse_linked_list(struct club *head)
193 {
194     struct club *current = head->next;
195     struct club *prev = NULL, *future = NULL;
196
197     while (current != NULL)
198     {
199         // Store next
200         future = current->next;
201         // Reverse current node's pointer
202         current->next = prev;
203         // Move pointers one position ahead.
204         prev = current;
205         current = future;
206     }
207     head->next = prev;
208 }
209
210 int findLength(struct club *head)
211 {
212     int i = 0;
213     struct club *temp = head;
214     while (temp != NULL)
215     {
216         temp = temp->next;
217         i++;
218     }
219     return i;
220 }
221
222 int insertByPosition(struct club *head)
223 {
224     int i = 1, pos = 0;
225     struct club *curr = head;
226     struct club *nnode = (struct club *)malloc(sizeof(struct club));
227     printf("Enter the name of the person you want to add to the club: \n");
228     scanf("%s", &nnode->name);
229     printf("Enter the prn of the person you want to add to the club: \n");
230     scanf("%d", &nnode->prn);
231     printf("Enter the position of the person you want to add to the club: \n");
232     scanf("%d", &pos);
233     pos--;
234     int total_length = findLength(head);
235     if (pos > total_length + 1)
236     {
237         printf("Data cant be inserted!\n");
238         return 0;
239     }
240     else
241     {
242         while (curr != NULL && i < total_length)
243         {
244             i++;
245             curr = curr->next;
246         }
247         nnode->next = curr->next;
```

```
248     curr->next = nnode;
249 }
250 }
251
252 int main()
253 {
254     int choice = 0;
255     int eligible_year = 0;
256     printf("What Year do you belong to? (1, 2, 3, 4)\n\n");
257     scanf("%d", &eligible_year);
258     if (eligible_year < 2 || eligible_year > 4)
259     {
260         printf("\nYou arent eligible to join the club\n");
261         return 0;
262     }
263     // Creating the Club
264     struct club *head, *merged_head;
265     head = (struct club *)malloc(sizeof(struct club));
266     head->next = NULL;
267
268     // Creating the Club
269     struct club *head_2;
270     head_2 = (struct club *)malloc(sizeof(struct club));
271     head_2->next = NULL;
272
273     // Creating the Merged Club
274     merged_head = (struct club *)malloc(sizeof(struct club));
275     merged_head->next = NULL;
276     while (1)
277     {
278         printf("\nEnter What you want to do: \n\n\
279 1. Enroll to the club\n\
280 2. Delete a member of the Club (1 for president, 0 for Secretary)\n\
281 3. Find total members\n\
282 4. View List of Club Members\n\
283 5. Sorting The Members of the Club by PRN\n\
284 6. Merge 2 Clubs (First fill the 1st Club)\n\
285 7. Reverse Members of the Club\n\
286 8. Sort the Members of the Merged Club\n\
287 9. Insert an element in a position. \n\n\
288 ");
289
290         scanf("%d", &choice);
291         switch (choice)
292         {
293             case 1:
294                 add_member(head);
295                 display_club(head);
296                 break;
297             case 2:
298                 int delete_position;
299                 printf("What position do you want to delete the node? ");
300                 scanf("%d", &delete_position);
301                 delete_member(head, delete_position);
302                 display_club(head);
303                 break;
304             case 3:
305                 printf("\nThe Total Number of Elements in the Club is: %d\n",
find_total_members(head));
```

```
306         break;
307     case 4:
308         printf("\nThe Members of the Club are: \n");
309         display_club(head);
310         break;
311     case 5:
312         printf("\nThe Members of the Club Before Sorting by PRN Number are: \n
313 ");
314         display_club(head);
315         sort_linked_list(head);
316         printf("\nThe Members of the Club After Sorting by PRN Number are: \n
317 ");
318         display_club(head);
319         break;
320     case 6:
321         if (head->next == NULL)
322         {
323             printf("\nNothing in the First Club! Add data there first\n\n");
324             add_member(head);
325         }
326         printf("\nThe Members of the First Club Are: \n");
327         display_club(head);
328         printf("\nAdd the Members to the Second Club: \n");
329         add_member(head_2);
330         printf("\nThe Members of the Second Club Are: \n");
331         display_club(head_2);
332         merged_head = merge_2_linked_list(head, head_2);
333         printf("\nOn Combining the 2 Linked Lists: \n");
334         display_club(merged_head);
335         break;
336     case 7:
337         printf("The Members of the Club Before Reversing are: \n");
338         display_club(head);
339         reverse_linked_list(head);
340         printf("The Members of the Club After Reversing are: \n");
341         display_club(head);
342         break;
343     case 8:
344         printf("\nThe Members of the Merged Club Before Sorting by PRN Number
345 are: \n");
346         display_club(merged_head);
347         sort_linked_list(merged_head);
348         printf("\nThe Members of the Merged Club After Sorting by PRN Number
349 are: \n");
350         display_club(merged_head);
351         break;
352     case 9:
353         printf("Which position do you wanna add the new value?\n");
354         insertByPosition(head);
355     default:
356         break;
357 }
358 }
359 return 0;
360 }
```

Listing 1: Main.Cpp

### 8.3 Input and Output

## *FDS Assignment 5 - Singly Linked List Operations*

---

```
1 What Year do you belong to? (1, 2, 3, 4)
2
3 4
4
5 Enter What you want to do:
6
7     1. Enroll to the club
8     2. Delete a member of the Club (1 for president, 0 for Secretary)
9     3. Find total members
10    4. View List of Club Members
11    5. Sorting The Members of the Club by PRN
12    6. Merge 2 Clubs (First fill the 1st Club)
13    7. Reverse Members of the Club
14    8. Sort the Members of the Merged Club
15    9. Insert an element in a position.
16
17     1
18
19 Enter Name of the President: Presi
20
21 Enter the PRN: 109
22
23 Do you want to enter more Members? (1 or 0)1
24
25 Enter Name: Ramesh
26
27 Enter the PRN: 243
28
29 Do you want to enter more Members? (1 or 0)1
30
31 Enter Name: Suresh
32
33 Enter the PRN: 345
34
35 Do you want to enter more Members? (1 or 0)1
36
37 Enter Name: William
38
39 Enter the PRN: 99
40
41 Do you want to enter more Members? (1 or 0)1
42
43 Enter Name: Tony
44
45 Enter the PRN: 909
46
47 Do you want to enter more Members? (1 or 0)0
48
49 Enter The name of the Secretary: Pam
50
51 Enter the PRN: 005
52
53 --Member: 1--
54 Name: Presi
55 PRN: 109
56 --Member: 2--
57 Name: Ramesh
58 PRN: 243
59 --Member: 3--
```

## *FDS Assignment 5 - Singly Linked List Operations*

---

```
60 Name: Suresh
61 PRN: 345
62 --Member: 4--
63 Name: William
64 PRN: 99
65 --Member: 5--
66 Name: Tony
67 PRN: 909
68 --Member: 6--
69 Name: Pam
70 PRN: 5
71
72 Enter What you want to do:
73
74     1. Enroll to the club
75     2. Delete a member of the Club (1 for president, 0 for Secretary)
76     3. Find total members
77     4. View List of Club Members
78     5. Sorting The Members of the Club by PRN
79     6. Merge 2 Clubs (First fill the 1st Club)
80     7. Reverse Members of the Club
81     8. Sort the Members of the Merged Club
82     9. Insert an element in a position.
83
84     2
85 What position do you want to delete the node? 3
86
87 --Member: 1--
88 Name: Presi
89 PRN: 109
90 --Member: 2--
91 Name: Ramesh
92 PRN: 243
93 --Member: 3--
94 Name: William
95 PRN: 99
96 --Member: 4--
97 Name: Tony
98 PRN: 909
99 --Member: 5--
100 Name: Pam
101 PRN: 5
102
103
104 Enter What you want to do:
105
106     1. Enroll to the club
107     2. Delete a member of the Club (1 for president, 0 for Secretary)
108     3. Find total members
109     4. View List of Club Members
110     5. Sorting The Members of the Club by PRN
111     6. Merge 2 Clubs (First fill the 1st Club)
112     7. Reverse Members of the Club
113     8. Sort the Members of the Merged Club
114     9. Insert an element in a position.
115
116     5
117
118 The Members of the Club Before Sorting by PRN Number are:
```

## *FDS Assignment 5 - Singly Linked List Operations*

---

```
119
120 --Member: 1--
121 Name: Presi
122 PRN: 109
123 --Member: 2--
124 Name: Ramesh
125 PRN: 243
126 --Member: 3--
127 Name: William
128 PRN: 99
129 --Member: 4--
130 Name: Tony
131 PRN: 909
132 --Member: 5--
133 Name: Pam
134 PRN: 5
135
136 The Members of the Club After Sorting by PRN Number are:
137
138 --Member: 1--
139 Name: Pam
140 PRN: 5
141 --Member: 2--
142 Name: William
143 PRN: 99
144 --Member: 3--
145 Name: Presi
146 PRN: 109
147 --Member: 4--
148 Name: Ramesh
149 PRN: 243
150 --Member: 5--
151 Name: Tony
152 PRN: 909
153
154 Enter What you want to do:
155
156     1. Enroll to the club
157     2. Delete a member of the Club (1 for president, 0 for Secretary)
158     3. Find total members
159     4. View List of Club Members
160     5. Sorting The Members of the Club by PRN
161     6. Merge 2 Clubs (First fill the 1st Club)
162     7. Reverse Members of the Club
163     8. Sort the Members of the Merged Club
164     9. Insert an element in a position.
165
166     6
167
168 The Members of the First Club Are:
169
170 --Member: 1--
171 Name: Pam
172 PRN: 5
173 --Member: 2--
174 Name: William
175 PRN: 99
176 --Member: 3--
177 Name: Presi
```



## *FDS Assignment 5 - Singly Linked List Operations*

---

```
178 PRN: 109
179 --Member: 4--
180 Name: Ramesh
181 PRN: 243
182 --Member: 5--
183 Name: Tony
184 PRN: 909
185
186 Add the Members to the Second Club:
187
188 Enter Name of the President: Lewis
189
190 Enter the PRN: 75
191
192 Do you want to enter more Members? (1 or 0)1
193
194 Enter Name: Max
195
196 Enter the PRN: 1
197
198 Do you want to enter more Members? (1 or 0)1
199
200 Enter Name: Kevin
201
202 Enter the PRN: 665
203
204 Do you want to enter more Members? (1 or 0)1
205
206 Enter Name: Perez
207
208 Enter the PRN: 334
209
210 Do you want to enter more Members? (1 or 0)0
211
212 Enter The name of the Secretary: Erin
213
214 Enter the PRN: 443
215
216 The Members of the Second Club Are:
217
218 --Member: 1--
219 Name: Lewis
220 PRN: 75
221 --Member: 2--
222 Name: Max
223 PRN: 1
224 --Member: 3--
225 Name: Kevin
226 PRN: 665
227 --Member: 4--
228 Name: Perez
229 PRN: 334
230 --Member: 5--
231 Name: Erin
232 PRN: 443
233
234 On Combining the 2 Linked Lists:
235
236 --Member: 1--
```

## *FDS Assignment 5 - Singly Linked List Operations*

---

```
237 Name: Pam
238 PRN: 5
239 --Member: 2--
240 Name: William
241 PRN: 99
242 --Member: 3--
243 Name: Presi
244 PRN: 109
245 --Member: 4--
246 Name: Ramesh
247 PRN: 243
248 --Member: 5--
249 Name: Tony
250 PRN: 909
251 --Member: 6--
252 Name: Lewis
253 PRN: 75
254 --Member: 7--
255 Name: Max
256 PRN: 1
257 --Member: 8--
258 Name: Kevin
259 PRN: 665
260 --Member: 9--
261 Name: Perez
262 PRN: 334
263 --Member: 10--
264 Name: Erin
265 PRN: 443
266
267 Enter What you want to do:
268
269     1. Enroll to the club
270     2. Delete a member of the Club (1 for president, 0 for Secretary)
271     3. Find total members
272     4. View List of Club Members
273     5. Sorting The Members of the Club by PRN
274     6. Merge 2 Clubs (First fill the 1st Club)
275     7. Reverse Members of the Club
276     8. Sort the Members of the Merged Club
277     9. Insert an element in a position.
278
279     9
280 Which position do you wanna add the new value?
281 Enter the name of the person you want to add to the club:
282 Russell
283 Enter the prn of the person you want to add to the club:
284 345
285 Enter the position of the person you want to add to the club:
286 3
287
288 Enter What you want to do:
289
290     1. Enroll to the club
291     2. Delete a member of the Club (1 for president, 0 for Secretary)
292     3. Find total members
293     4. View List of Club Members
294     5. Sorting The Members of the Club by PRN
295     6. Merge 2 Clubs (First fill the 1st Club)
```

## *FDS Assignment 5 - Singly Linked List Operations*

---

```
296     7. Reverse Members of the Club
297     8. Sort the Members of the Merged Club
298     9. Insert an element in a position.
299
300     4
301
302 The Members of the Club are:
303
304 --Member: 1--
305 Name: Pam
306 PRN: 5
307 --Member: 2--
308 Name: William
309 PRN: 99
310 --Member: 3--
311 Name: Presi
312 PRN: 109
313 --Member: 4--
314 Name: Ramesh
315 PRN: 243
316 --Member: 5--
317 Name: Tony
318 PRN: 909
319 --Member: 6--
320 Name: Russell
321 PRN: 345
322
323 Enter What you want to do:
324
325     1. Enroll to the club
326     2. Delete a member of the Club (1 for president, 0 for Secretary)
327     3. Find total members
328     4. View List of Club Members
329     5. Sorting The Members of the Club by PRN
330     6. Merge 2 Clubs (First fill the 1st Club)
331     7. Reverse Members of the Club
332     8. Sort the Members of the Merged Club
333     9. Insert an element in a position.
334
335     7
336 The Members of the Club Before Reversing are:
337
338 --Member: 1--
339 Name: Pam
340 PRN: 5
341 --Member: 2--
342 Name: William
343 PRN: 99
344 --Member: 3--
345 Name: Presi
346 PRN: 109
347 --Member: 4--
348 Name: Ramesh
349 PRN: 243
350 --Member: 5--
351 Name: Tony
352 PRN: 909
353 --Member: 6--
354 Name: Russell
```

## *FDS Assignment 5 - Singly Linked List Operations*

---

```
355 PRN: 345
356 The Members of the Club After Reversing are:
357
358 --Member: 1--
359 Name: Russell
360 PRN: 345
361 --Member: 2--
362 Name: Tony
363 PRN: 909
364 --Member: 3--
365 Name: Ramesh
366 PRN: 243
367 --Member: 4--
368 Name: Presi
369 PRN: 109
370 --Member: 5--
371 Name: William
372 PRN: 99
373 --Member: 6--
374 Name: Pam
375 PRN: 5
376
377
378 Enter What you want to do:
379
380     1. Enroll to the club
381     2. Delete a member of the Club (1 for president, 0 for Secretary)
382     3. Find total members
383     4. View List of Club Members
384     5. Sorting The Members of the Club by PRN
385     6. Merge 2 Clubs (First fill the 1st Club)
386     7. Reverse Members of the Club
387     8. Sort the Members of the Merged Club
388     9. Insert an element in a position.
389
390     8
391
392 The Members of the Merged Club Before Sorting by PRN Number are:
393
394 --Member: 1--
395 Name: Pam
396 PRN: 5
397 --Member: 2--
398 Name: William
399 PRN: 99
400 --Member: 3--
401 Name: Presi
402 PRN: 109
403 --Member: 4--
404 Name: Ramesh
405 PRN: 243
406 --Member: 5--
407 Name: Tony
408 PRN: 909
409 --Member: 6--
410 Name: Lewis
411 PRN: 75
412 --Member: 7--
413 Name: Max
```

```
414 PRN: 1
415 --Member: 8--
416 Name: Kevin
417 PRN: 665
418 --Member: 9--
419 Name: Perez
420 PRN: 334
421 --Member: 10--
422 Name: Erin
423 PRN: 443
424
425 The Members of the Merged Club After Sorting by PRN Number are:
426
427 --Member: 1--
428 Name: Max
429 PRN: 1
430 --Member: 2--
431 Name: Pam
432 PRN: 5
433 --Member: 3--
434 Name: William
435 PRN: 99
436 --Member: 4--
437 Name: Lewis
438 PRN: 75
439 --Member: 5--
440 Name: Presi
441 PRN: 109
442 --Member: 6--
443 Name: Ramesh
444 PRN: 243
445 --Member: 7--
446 Name: Perez
447 PRN: 334
448 --Member: 8--
449 Name: Erin
450 PRN: 443
451 --Member: 9--
452 Name: Kevin
453 PRN: 665
454 --Member: 10--
455 Name: Tony
456 PRN: 909
```

Listing 2: Output

## **9 Time Complexity**

The Time complexities for Each Operation:

1. Creation of a Single Linked List:

$O(1)$

2. Insertion a Node at the End of a Single Linked List:

$O(1)$

3. Insertion a Node at A Random Position of a Single Linked List:

$O(N), \Omega(1)$

4. Accessing a Node at A Random Position of a Single Linked List:

$$O(N), \Omega(1)$$

5. Deletion of a Node at the End of a Single Linked List:

$$O(1)$$

6. Deletion at a Random Position of a Single Linked List:

$$O(N), \Omega(1)$$

## 10 Conclusion

Thus, implemented all the operations of an Abstract Data type like Inserting, Searching, Sorting, Deleting and Reversing on a Singly Linked List.

## 11 FAQs

1. **Write an ADT for Singly Linked List.**

An Abstract Data Type has 4 Main Operations:

- (a) Pseudo Code for Insertion in a Singly Linked List

```
1 struct node *curr = head;
2 struct node *nnode = (struct node *)malloc(sizeof(struct node));
3 print(Enter data)
4 scan(pos)
5 pos--;
6 int total_length = findLength(head);
7 if (pos > total_length + 1)
8     printf("Data cant be inserted!\n");
9     return 0;
10 else
11     while (curr != NULL && i < total_length)
12         i++;
13         curr = curr->next;
14     nnode->next = curr->next;
15     curr->next = nnode;
16
```

- (b) Pseudo Code for Deletion in a Singly Linked List

```
1 if (head->next == NULL)
2     print("list empty")
3     return -1;
4 struct node *prev = (struct node *)malloc(sizeof(struct node *));
5 prev = head;
6 int count = 1;
7 struct node *curr = (struct node *)malloc(sizeof(struct node *));
8 curr = head->next;
9
10 int len = find_total_members(head);
11
12 if (position > len)
```

```
13     scanf("\nCant Delete data");
14 else
15     while (count < position && curr->next != NULL)
16         count++;
17         prev = curr;
18         curr = curr->next;
19 struct node *temp = (struct node *)malloc(sizeof(struct node *));
20
21 temp = curr;
22 prev->next = curr->next;
23 curr->next = NULL;
24 free(temp);
25
```

### (c) Pseudo Code for Searching an Element in a Single Linked List

```
1 printf("\nEnter item which you want to search?\n");
2 scanf("%d",&item);
3 while (ptr!=NULL)
4     if(ptr->data == item)
5         printf("item found at location %d ",i+1);
6         flag=0;
7     else
8         flag=1;
9     i++;
10    ptr = ptr -> next;
11 if(flag==1)
12     printf("Item not found\n");
13
```

### (d) Pseudo Code for Modifying an Element in a Single Linked List

```
1 printf("\nEnter item which you want to Modify?\n");
2 scanf("%d",&item);
3 printf("\nEnter Modified data?\n");
4 scanf("%d",&data_new);
5 while (ptr!=NULL)
6     if(ptr->data == item)
7         ptr->data = data_new;
8         printf(item modified);
9         flag=0;
10    else
11        flag=1;
12    i++;
13    ptr = ptr -> next;
14 if(flag==1)
15     printf("Item not found\n");
16
```

## 2. What are the disadvantages of a Singly Linked List?

- (a) *Memory usage* : More memory is required in the linked list as compared to an array. Because in a linked list, a pointer is also required to store the address of the next element and it requires extra memory for itself.
- (b) *Traversal*: In a Linked list traversal is more time-consuming as compared to an array. Direct access to an element is not possible in a linked list as in an array by index. For example, for accessing a node at position n, one has to traverse all the nodes before it.

- (c) *Reverse Traversing*: In a singly linked list reverse traversing is not possible, but in the case of a doubly-linked list, it can be possible as it contains a pointer to the previously connected nodes with each node. For performing this extra memory is required for the back pointer hence, there is a wastage of memory.
- (d) *Random Access*: Random access is not possible in a linked list due to its dynamic memory allocation.

**3. Applications of Singly Linked List.**

- (a) Implementation of stacks and queues
- (b) Implementation of graphs: Adjacency list representation of graphs is the most popular which uses a linked list to store adjacent vertices.
- (c) Dynamic memory allocation: We use a linked list of free blocks.
- (d) Maintaining a directory of names
- (e) Performing arithmetic operations on long integers
- (f) Manipulation of polynomials by storing constants in the node of the linked list
- (g) representing sparse matrices