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

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

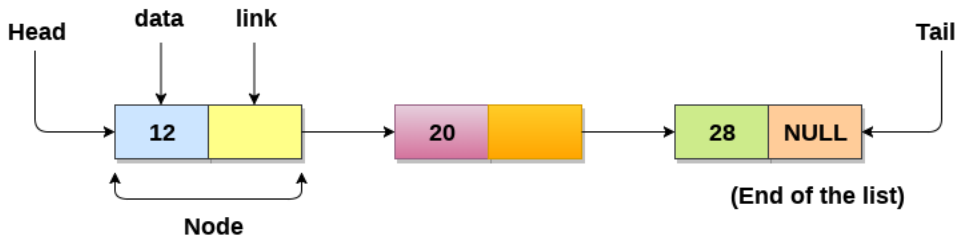
Department of Computer Engineering has student's node named 'Pinnacle Node'. Students of second, third and final year of department can be granted membership on request. Similarly, one may cancel the membership of node. First node is reserved for president of node and last node is reserved for the secretary 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 funtions makes it space efficient as opposed to passing the entire linked list or an entire array.
- The Head node being using 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 *Single 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 strucutre.
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 lits.
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. Displaing, 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

```
struct node *head, *merged_head;
head = (struct node *)malloc(sizeof(struct node));
head->next = NULL;
// then add new nodes.
struct node *temp = head;
while(enter elements)
  struct node *curr = (struct node *)malloc(sizeof(struct node));
  scanf("%s", curr->Data);
  curr->next = NULL;
  temp->next = curr;
  temp = curr;
```

8.1.2 Pseudo Code for Display of a Singly Linked List

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

8.1.3 Pseudo Code for Insertion in a Singly Linked List

```
struct node *curr = head;
  struct node *nnode = (struct node *)malloc(sizeof(struct node));
  print(Enter data)
scan(pos)
  pos--;
  int total_length = findLength(head);
```

```
    if (pos > total_length + 1)
        printf("Data cant be inserted!\n");
        return 0;
    else
        while (curr != NULL && i < total_length)
            i++;
            curr = curr->next;
        nnode->next = curr->next;
        curr->next = nnode;
```

8.1.4 Pseudo Code for Deletion in a Singly Linked List

```
 if (head->next == NULL)
   print("list empty")
   return -1;
   struct node *prev = (struct node *)malloc(sizeof(struct node *));
   prev = head;
   int count = 1;
   struct node *curr = (struct node *)malloc(sizeof(struct node *));
   curr = head->next;

   int len = find_total_members(head);

   if (position > len)
       scanf("\nCant Delete data");
   else
       while (count < position && curr->next != NULL)
           count++;
           prev = curr;
           curr = curr->next;
   struct node *temp = (struct node *)malloc(sizeof(struct node *));

   temp = curr;
   prev->next = curr->next;
   curr->next = NULL;
   free(temp);
```

8.1.5 Pseudo Code for Reversing of a Singly Linked List

```
    struct node *current = head->next;
    struct node *prev = NULL, *future = NULL;
    while (current != NULL)
        future = current->next;
        current->next = prev;
        prev = current;
        current = future;
    head->next = prev;
```

8.1.6 Pseudo Code for Sorting of a Singly Linked List

```
 struct node *i = (struct node *)malloc(sizeof(struct node));
   struct node *j = (struct node *)malloc(sizeof(struct node));
   struct node temp;

   // bubble sorting
```

```
    for (i = head; i != NULL; i = i->next)
        for (j = i; j->next != NULL; j = j->next)
            if (j->prn > j->next->prn)
                temp.prn = j->prn;
                j->prn = j->next->prn;
                j->next->prn = temp.prn;
```

8.1.7 Pseudo Code for Merging of 2 Singly Linked Lists

```
  struct node *merged_head = (struct node *)malloc(sizeof(struct node));
    merged_head->next = NULL;
    if (head->next == NULL || head_2->next == NULL)
        printf("\nOne of the nodes is empty, so no point in merging!\n");
        return merged_head;

    struct node *temp_head, *temp_merged, *current;
    temp_head = head->next;

    temp_merged = (struct node *)malloc(sizeof(struct node));
    merged_head->next = temp_merged;
    while (temp_head != NULL)
        current = temp_merged;
        temp_merged->prn = temp_head->prn;

        temp_merged = (struct node *)malloc(sizeof(struct node));
        current->next = temp_merged;
        temp_head = temp_head->next;

    temp_head = head_2->next;
    while (temp_head != NULL)
        current = temp_merged;
        temp_merged->prn = temp_head->prn;

        temp_merged = (struct node *)malloc(sizeof(struct node));
        current->next = temp_merged;
        temp_head = temp_head->next;

    current->next = NULL;

    return merged_head;
```

8.2 C Implementation of Problem Statement

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
// something about trhe president and the club.
// First and last are reserved for the president and the secretary.
// people can enroll in the club as well.

struct club
{
    int prn;
    char name[30];
    struct club *next;
};
```

```
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;
```

```
    int count = 1;
    struct club *curr = (struct club *)malloc(sizeof(struct club *));

    curr = head->next;

    int len = find_total_members(head);
    if (position > len)
    {
        scanf("\nCant Delete data");
    }
    else
    {
        while (count < position && curr->next != NULL)
        {
            count++;
            prev = curr;
            curr = curr->next;
        }
    }
    struct club *temp = (struct club *)malloc(sizeof(struct club *));

    temp = curr;
    prev->next = curr->next;
    curr->next = NULL;
    free(temp);
}

int display_club(struct club *head)
{
    int counter = 0;
    if (head->next == NULL)
    {
        printf("\nNo members in the club");
        return -1;
    }
    struct club *curr = (struct club *)malloc(sizeof(struct club));
    curr = head->next;
    while (curr != NULL)
    {
        printf("\n--Member: %d--", ++counter);
        printf("\nName: %s", curr->name);
        printf("\nPRN: %d", curr->prn);
        curr = curr->next;
    }
    printf("\n");
}

void sort_linked_list(struct club *head)
{
    struct club *i = (struct club *)malloc(sizeof(struct club));
    struct club *j = (struct club *)malloc(sizeof(struct club));
    struct club temp;

    // bubble sorting
    for (i = head; i != NULL; i = i->next)
    {
        for (j = i; j->next != NULL; j = j->next)
        {
            if (j->prn > j->next->prn)
```

```
            {
                // swapping elements of a linked list here, lengthy but with the conventional logic.
                // we dont touch the next variable, coz that would change the order of the addresses and mess up the linked list.
                // so we just swap the actual values.
                temp.prn = j->prn;
                strcpy(temp.name, j->name);

                j->prn = j->next->prn;
                strcpy(j->name, j->next->name);

                j->next->prn = temp.prn;
                strcpy(j->next->name, temp.name);
            }
        }
    }
}

struct club *merge_2_linked_list(struct club *head, struct club *head_2)
{
    struct club *merged_head = (struct club *)malloc(sizeof(struct club));
    merged_head->next = NULL;
    if (head->next == NULL || head_2->next == NULL)
    {
        printf("\nOne of the clubs is empty, so no point in merging!\n");
        return merged_head;
    }

    struct club *temp_head, *temp_merged, *current;
    temp_head = head->next;

    temp_merged = (struct club *)malloc(sizeof(struct club));
    merged_head->next = temp_merged;
    while (temp_head != NULL)
    {
        current = temp_merged;
        strcpy(temp_merged->name, temp_head->name);
        temp_merged->prn = temp_head->prn;

        temp_merged = (struct club *)malloc(sizeof(struct club));
        current->next = temp_merged;
        temp_head = temp_head->next;
    }

    temp_head = head_2->next;
    while (temp_head != NULL)
    {
        current = temp_merged;
        strcpy(temp_merged->name, temp_head->name);
        temp_merged->prn = temp_head->prn;

        temp_merged = (struct club *)malloc(sizeof(struct club));
        current->next = temp_merged;
        temp_head = temp_head->next;
    }

    current->next = NULL;

```

```
    return merged_head;
}

void reverse_linked_list(struct club *head)
{
    struct club *current = head->next;
    struct club *prev = NULL, *future = NULL;

    while (current != NULL)
    {
        // Store next
        future = current->next;
        // Reverse current node's pointer
        current->next = prev;
        // Move pointers one position ahead.
        prev = current;
        current = future;
    }
    head->next = prev;
}

int findLength(struct club *head)
{
    int i = 0;
    struct club *temp = head;
    while (temp != NULL)
    {
        temp = temp->next;
        i++;
    }
    return i;
}

int insertByPosition(struct club *head)
{
    int i = 1, pos = 0;
    struct club *curr = head;
    struct club *nnode = (struct club *)malloc(sizeof(struct club));
    printf("Enter the name of the person you want to add to the club: \n");
    scanf("%s", &nnode->name);
    printf("Enter the prn of the person you want to add to the club: \n");
    scanf("%d", &nnode->prn);
    printf("Enter the position of the person you want to add to the club: \n");
    scanf("%d", &pos);
    pos--;
    int total_length = findLength(head);
    if (pos > total_length + 1)
    {
        printf("Data cant be inserted!\n");
        return 0;
    }
    else
    {
        while (curr != NULL && i < total_length)
        {
            i++;
            curr = curr->next;
        }
        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             display_club(head);
314             sort_linked_list(head);
315             printf("\nThe Members of the Club After Sorting by PRN Number are: \n"
    );
316             display_club(head);
317             break;
318         case 6:
319             if (head->next == NULL)
320             {
321                 printf("\nNothing in the First Club! Add data there first\n\n");
322                 add_member(head);
323             }
324             printf("\nThe Members of the First Club Are: \n");
325             display_club(head);
326             printf("\nAdd the Members to the Second Club: \n");
327             add_member(head_2);
328             printf("\nThe Members of the Second Club Are: \n");
329             display_club(head_2);
330             merged_head = merge_2_linked_list(head, head_2);
331             printf("\nOn Combining the 2 Linked Lists: \n");
332             display_club(merged_head);
333             break;
334         case 7:
335             printf("The Members of the Club Before Reversing are: \n");
336             display_club(head);
337             reverse_linked_list(head);
338             printf("The Members of the Club After Reversing are: \n");
339             display_club(head);
340             break;
341         case 8:
342             printf("\nThe Members of the Merged Club Before Sorting by PRN Number
    are: \n");
343             display_club(merged_head);
344             sort_linked_list(merged_head);
345             printf("\nThe Members of the Merged Club After Sorting by PRN Number
    are: \n");
346             display_club(merged_head);
347             break;
348         case 9:
349             printf("Which position do you wanna add the new value?\n");
350             insertByPosition(head);
351         default:
352             break;
353         };
354     }
355     return 0;
356 }
```

Listing 1: Main.Cpp

8.3 Input and Output

```
What Year do you belong to? (1, 2, 3, 4)

4

Enter What you want to do:

    1. Enroll to the club
    2. Delete a member of the Club (1 for president, 0 for Secretary)
    3. Find total members
    4. View List of Club Members
    5. Sorting The Members of the Club by PRN
    6. Merge 2 Clubs (First fill the 1st Club)
    7. Reverse Members of the Club
    8. Sort the Members of the Merged Club
    9. Insert an element in a position.

    1

Enter Name of the President: Presi

Enter the PRN: 109

Do you want to enter more Members? (1 or 0)1

Enter Name: Ramesh

Enter the PRN: 243

Do you want to enter more Members? (1 or 0)1

Enter Name: Suresh

Enter the PRN: 345

Do you want to enter more Members? (1 or 0)1

Enter Name: William

Enter the PRN: 99

Do you want to enter more Members? (1 or 0)1

Enter Name: Tony

Enter the PRN: 909

Do you want to enter more Members? (1 or 0)0

Enter The name of the Secretary: Pam

Enter the PRN: 005

--Member: 1--
Name: Presi
PRN: 109
--Member: 2--
Name: Ramesh
PRN: 243
--Member: 3--
```

```
Name: Suresh
PRN: 345
--Member: 4--
Name: William
PRN: 99
--Member: 5--
Name: Tony
PRN: 909
--Member: 6--
Name: Pam
PRN: 5

Enter What you want to do:

    1. Enroll to the club
    2. Delete a member of the Club (1 for president, 0 for Secretary)
    3. Find total members
    4. View List of Club Members
    5. Sorting The Members of the Club by PRN
    6. Merge 2 Clubs (First fill the 1st Club)
    7. Reverse Members of the Club
    8. Sort the Members of the Merged Club
    9. Insert an element in a position.

    2
What position do you want to delete the node? 3

--Member: 1--
Name: Presi
PRN: 109
--Member: 2--
Name: Ramesh
PRN: 243
--Member: 3--
Name: William
PRN: 99
--Member: 4--
Name: Tony
PRN: 909
--Member: 5--
Name: Pam
PRN: 5


Enter What you want to do:

    1. Enroll to the club
    2. Delete a member of the Club (1 for president, 0 for Secretary)
    3. Find total members
    4. View List of Club Members
    5. Sorting The Members of the Club by PRN
    6. Merge 2 Clubs (First fill the 1st Club)
    7. Reverse Members of the Club
    8. Sort the Members of the Merged Club
    9. Insert an element in a position.

    5

The Members of the Club Before Sorting by PRN Number are:
```

```
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
```

```
PRN: 109
--Member: 4--
Name: Ramesh
PRN: 243
--Member: 5--
Name: Tony
PRN: 909

Add the Members to the Second Club:

Enter Name of the President: Lewis

Enter the PRN: 75

Do you want to enter more Members? (1 or 0)1

Enter Name: Max

Enter the PRN: 1

Do you want to enter more Members? (1 or 0)1

Enter Name: Kevin

Enter the PRN: 665

Do you want to enter more Members? (1 or 0)1

Enter Name: Perez

Enter the PRN: 334

Do you want to enter more Members? (1 or 0)0

Enter The name of the Secretary: Erin

Enter the PRN: 443

The Members of the Second Club Are:

--Member: 1--
Name: Lewis
PRN: 75
--Member: 2--
Name: Max
PRN: 1
--Member: 3--
Name: Kevin
PRN: 665
--Member: 4--
Name: Perez
PRN: 334
--Member: 5--
Name: Erin
PRN: 443

On Combining the 2 Linked Lists:

--Member: 1--
```

```
Name: Pam
PRN: 5
--Member: 2--
Name: William
PRN: 99
--Member: 3--
Name: Presi
PRN: 109
--Member: 4--
Name: Ramesh
PRN: 243
--Member: 5--
Name: Tony
PRN: 909
--Member: 6--
Name: Lewis
PRN: 75
--Member: 7--
Name: Max
PRN: 1
--Member: 8--
Name: Kevin
PRN: 665
--Member: 9--
Name: Perez
PRN: 334
--Member: 10--
Name: Erin
PRN: 443

Enter What you want to do:

    1. Enroll to the club
    2. Delete a member of the Club (1 for president, 0 for Secretary)
    3. Find total members
    4. View List of Club Members
    5. Sorting The Members of the Club by PRN
    6. Merge 2 Clubs (First fill the 1st Club)
    7. Reverse Members of the Club
    8. Sort the Members of the Merged Club
    9. Insert an element in a position.

    9
Which position do you wanna add the new value?
Enter the name of the person you want to add to the club:
Russell
Enter the prn of the person you want to add to the club:
345
Enter the position of the person you want to add to the club:
3

Enter What you want to do:

    1. Enroll to the club
    2. Delete a member of the Club (1 for president, 0 for Secretary)
    3. Find total members
    4. View List of Club Members
    5. Sorting The Members of the Club by PRN
    6. Merge 2 Clubs (First fill the 1st Club)
```

```
    7. Reverse Members of the Club
    8. Sort the Members of the Merged Club
    9. Insert an element in a position.

    4

The Members of the Club are:

--Member: 1--
Name: Pam
PRN: 5
--Member: 2--
Name: William
PRN: 99
--Member: 3--
Name: Presi
PRN: 109
--Member: 4--
Name: Ramesh
PRN: 243
--Member: 5--
Name: Tony
PRN: 909
--Member: 6--
Name: Russell
PRN: 345

Enter What you want to do:

    1. Enroll to the club
    2. Delete a member of the Club (1 for president, 0 for Secretary)
    3. Find total members
    4. View List of Club Members
    5. Sorting The Members of the Club by PRN
    6. Merge 2 Clubs (First fill the 1st Club)
    7. Reverse Members of the Club
    8. Sort the Members of the Merged Club
    9. Insert an element in a position.

    7
The Members of the Club Before Reversing are:

--Member: 1--
Name: Pam
PRN: 5
--Member: 2--
Name: William
PRN: 99
--Member: 3--
Name: Presi
PRN: 109
--Member: 4--
Name: Ramesh
PRN: 243
--Member: 5--
Name: Tony
PRN: 909
--Member: 6--
Name: Russell
```

```
PRN: 345
The Members of the Club After Reversing are:

--Member: 1--
Name: Russell
PRN: 345
--Member: 2--
Name: Tony
PRN: 909
--Member: 3--
Name: Ramesh
PRN: 243
--Member: 4--
Name: Presi
PRN: 109
--Member: 5--
Name: William
PRN: 99
--Member: 6--
Name: Pam
PRN: 5


Enter What you want to do:

    1. Enroll to the club
    2. Delete a member of the Club (1 for president, 0 for Secretary)
    3. Find total members
    4. View List of Club Members
    5. Sorting The Members of the Club by PRN
    6. Merge 2 Clubs (First fill the 1st Club)
    7. Reverse Members of the Club
    8. Sort the Members of the Merged Club
    9. Insert an element in a position.

    8

The Members of the Merged Club Before Sorting by PRN Number are:

--Member: 1--
Name: Pam
PRN: 5
--Member: 2--
Name: William
PRN: 99
--Member: 3--
Name: Presi
PRN: 109
--Member: 4--
Name: Ramesh
PRN: 243
--Member: 5--
Name: Tony
PRN: 909
--Member: 6--
Name: Lewis
PRN: 75
--Member: 7--
Name: Max
```

```
PRN: 1
--Member: 8--
Name: Kevin
PRN: 665
--Member: 9--
Name: Perez
PRN: 334
--Member: 10--
Name: Erin
PRN: 443

The Members of the Merged Club After Sorting by PRN Number are:

--Member: 1--
Name: Max
PRN: 1
--Member: 2--
Name: Pam
PRN: 5
--Member: 3--
Name: William
PRN: 99
--Member: 4--
Name: Lewis
PRN: 75
--Member: 5--
Name: Presi
PRN: 109
--Member: 6--
Name: Ramesh
PRN: 243
--Member: 7--
Name: Perez
PRN: 334
--Member: 8--
Name: Erin
PRN: 443
--Member: 9--
Name: Kevin
PRN: 665
--Member: 10--
Name: Tony
PRN: 909
```

Listing 2: Output

9 Time Complexity

The Time complexities for Each Opertion:

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

```
struct node *curr = head;
struct node *nnode = (struct node *)malloc(sizeof(struct node));
print(Enter data)
scan(pos)
pos--;
int total_length = findLength(head);
if (pos > total_length + 1)
  printf("Data cant be inserted!\n");
  return 0;
else
  while (curr != NULL && i < total_length)
    i++;
    curr = curr->next;
  nnode->next = curr->next;
  curr->next = nnode;

```

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

```
if (head->next == NULL)
  print("list empty")
  return -1;
struct node *prev = (struct node *)malloc(sizeof(struct node *));
prev = head;
int count = 1;
struct node *curr = (struct node *)malloc(sizeof(struct node *));
curr = head->next;

int len = find_total_members(head);

if (position > len)
```

```
  scanf("\nCant Delete data");
else
  while (count < position && curr->next != NULL)
    count++;
    prev = curr;
    curr = curr->next;
struct node *temp = (struct node *)malloc(sizeof(struct node *));

temp = curr;
prev->next = curr->next;
curr->next = NULL;
free(temp);

```

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

```
printf("\nEnter item which you want to search?\n");
scanf("%d",&item);
while (ptr!=NULL)
  if(ptr->data == item)
    printf("item found at location %d ",i+1);
    flag=0;
  else
    flag=1;
  i++;
  ptr = ptr -> next;
if(flag==1)
  printf("Item not found\n");

```

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

```
printf("\nEnter item which you want to Modify?\n");
scanf("%d",&item);
printf("\nEnter Modified data?\n");
scanf("%d",&data_new);
while (ptr!=NULL)
  if(ptr->data == item)
    ptr->data = data_new;
    printf(item modified);
    flag=0;
  else
    flag=1;
  i++;
  ptr = ptr -> next;
if(flag==1)
  printf("Item not found\n");

```

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