**Lab-2: 19CSE212 Data Structure and Algorithms**

**Implementation of Single Linked List Data Structure**

**Node Creation**

1. struct node
2. {
3. **int** data;
4. struct node \*next;
5. };
6. struct node \*head, \*ptr;
7. ptr = (struct node \*)malloc(sizeof(struct node \*));

**C Program to Implement Insertion, Deletion Operations**

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

**// Node Creation**

1. struct node
2. {
3. **int** data;
4. struct node \*next;
5. };
6. struct node \*head;  // **pointing to the start node of the linked list**

1. **void** beginsert ();   // inserting an element at the beginning of the linked list
2. **void** lastinsert ();   // inserting an element at the last of the linked list
3. **void** randominsert();  // inserting at any specified position of the linked list
4. **void** begin\_delete();  // deleting from the beginning of the linked list
5. **void** last\_delete();   // deleting an element from the last of the linked list
6. **void** random\_delete();  // inserting at any specified position of the linked list
7. **void** display();   // displaying the element of the linked list
8. **void** search();   // searching an element in the linked list
9. **void** main ()
10. {
11. **int** choice =0;
12. **while**(choice != 9)
13. {
14. printf("\n\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");
15. printf("\nChoose one option from the following list ...\n");
16. printf("\n===============================================\n");
17. printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random loc ation\n4.Delete from Beginning\n
18. 5.Delete from last\n6.Delete node after specified location\n7.Search **for** an element\n8.Show\n9.Exit\n");
19. printf("\nEnter your choice?\n");
20. scanf("\n%d",&choice);
21. **switch**(choice)
22. {
23. **case** 1:
24. beginsert();
25. **break**;
26. **case** 2:
27. lastinsert();
28. **break**;
29. **case** 3:
30. randominsert();
31. **break**;
32. **case** 4:
33. begin\_delete();
34. **break**;
35. **case** 5:
36. last\_delete();
37. **break**;
38. **case** 6:
39. random\_delete();
40. **break**;
41. **case** 7:
42. search();
43. **break**;
44. **case** 8:
45. display();
46. **break**;
47. **case** 9:
48. exit(0);
49. **break**;
50. **default**:
51. printf("Please enter valid choice..");
52. }
53. }
54. }

**// inserting at the beginning**

1. **void** beginsert()
2. {
3. struct node \*ptr;
4. **int** item;
5. ptr = (struct node \*) malloc(sizeof(struct node \*));
6. **if**(ptr == NULL)
7. {
8. printf("\nOVERFLOW");
9. }
10. **else**
11. {
12. printf("\nEnter value\n");
13. scanf("%d",&item);
14. ptr->data = item;
15. ptr->next = head;
16. head = ptr;
17. printf("\nNode inserted");
18. }
20. }

**// inserting at the End**

1. **void** lastinsert()
2. {
3. struct node \*ptr,\*temp;
4. **int** item;
5. ptr = (struct node\*)malloc(sizeof(struct node));
6. **if**(ptr == NULL)
7. {
8. printf("\nOVERFLOW");
9. }
10. **else**
11. {
12. printf("\nEnter value?\n");
13. scanf("%d",&item);
14. ptr->data = item;
15. **if**(head == NULL)
16. {
17. ptr -> next = NULL;
18. head = ptr;
19. printf("\nNode inserted");
20. }
21. **else**
22. {
23. temp = head;
24. **while** (temp -> next != NULL)
25. {
26. temp = temp -> next;
27. }
28. temp->next = ptr;
29. ptr->next = NULL;
30. printf("\nNode inserted");
32. }
33. }
34. }

**// inserting at any specified position**

1. **void** randominsert()
2. {
3. **int** i,loc,item;
4. struct node \*ptr, \*temp;
5. ptr = (struct node \*) malloc (sizeof(struct node));
6. **if**(ptr == NULL)
7. {
8. printf("\nOVERFLOW");
9. }
10. **else**
11. {
12. printf("\nEnter element value");
13. scanf("%d",&item);
14. ptr->data = item;
15. printf("\nEnter the location after which you want to insert ");
16. scanf("\n%d",&loc);
17. temp=head;
18. **for**(i=0;i<loc;i++)
19. {
20. temp = temp->next;
21. **if**(temp == NULL)
22. {
23. printf("\ncan't insert\n");
24. **return**;
25. }
27. }
28. ptr ->next = temp ->next;
29. temp ->next = ptr;
30. printf("\nNode inserted");
31. }
32. }

**// Deleting from the beginning**

1. **void** begin\_delete()
2. {
3. struct node \*ptr;
4. **if**(head == NULL)
5. {
6. printf("\nList is empty\n");
7. }
8. **else**
9. {
10. ptr = head;
11. head = ptr->next;
12. free(ptr);
13. printf("\nNode deleted from the begining ...\n");
14. }
15. }

**// Deleting from the last**

1. **void** last\_delete()
2. {
3. struct node \*ptr,\*ptr1;
4. **if**(head == NULL)
5. {
6. printf("\nlist is empty");
7. }
8. **else** **if**(head -> next == NULL)
9. {
10. head = NULL;
11. free(head);
12. printf("\nOnly node of the list deleted ...\n");
13. }
15. **else**
16. {
17. ptr = head;
18. **while**(ptr->next != NULL)
19. {
20. ptr1 = ptr;
21. ptr = ptr ->next;
22. }
23. ptr1->next = NULL;
24. free(ptr);
25. printf("\nDeleted Node from the last ...\n");
26. }
27. }

**// Deleting from any specified position**

1. **void** random\_delete()
2. {
3. struct node \*ptr,\*ptr1;
4. **int** loc,i;
5. printf("\n Enter the location of the node after which you want to perform deletion \n");
6. scanf("%d",&loc);
7. ptr=head;
8. **for**(i=0;i<loc;i++)
9. {
10. ptr1 = ptr;
11. ptr = ptr->next;
13. **if**(ptr == NULL)
14. {
15. printf("\nCan't delete");
16. **return**;
17. }
18. }
19. ptr1 ->next = ptr ->next;
20. free(ptr);
21. printf("\nDeleted node %d ",loc+1);
22. }

**// Searching an element in the list**

1. **void** search()
2. {
3. struct node \*ptr;
4. **int** item,i=0,flag;
5. ptr = head;
6. **if**(ptr == NULL)
7. {
8. printf("\nEmpty List\n");
9. }
10. **else**
11. {
12. printf("\nEnter item which you want to search?\n");
13. scanf("%d",&item);
14. **while** (ptr!=NULL)
15. {
16. **if**(ptr->data == item)
17. {
18. printf("item found at location %d ",i+1);
19. flag=0;
20. }
21. **else**
22. {
23. flag=1;
24. }
25. i++;
26. ptr = ptr -> next;
27. }
28. **if**(flag==1)
29. {
30. printf("Item not found\n");
31. }
32. }
34. }

**// Displaying elements of the Linked List**

1. **void** display()
2. {
3. struct node \*ptr;
4. ptr = head;
5. **if**(ptr == NULL)
6. {
7. printf("Nothing to print");
8. }
9. **else**
10. {
11. printf("\nprinting values . . . . .\n");
12. **while** (ptr!=NULL)
13. {
14. printf("\n%d",ptr->data);
15. ptr = ptr -> next;
16. }
17. }
18. }

**Assignments**

* 1. Write the pseudocode for reversing a single linked list and implement the same.
  2. Implement insertion sort using a single linked list.
  3. Check if a given single linked list is a palindrome or not.