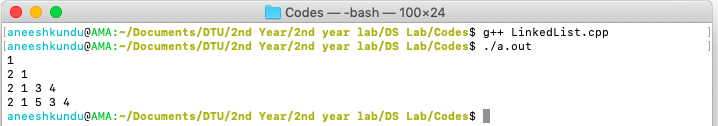
**Experiment 7**

**AIM:** Write a program to implement a linked list which supports insertion at head, tail, and at an index.

**THEORY:** Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.

**Code:**

1. #include < iostream >
2. using namespace std;
3. struct Node {
4. int data;
5. Node \* next;
6. Node() {
7. data = 0;
8. next = NULL;
9. }
10. Node(int d) {
11. data = d;
12. next = NULL;
13. }
14. };
15. struct LinkedList {
16. Node \* head;
17. Node \* tail;
18. LinkedList() {
19. head = tail = NULL;
20. }
21. void insert\_at\_tail(int v) {
22. if (head == NULL) {
23. head = new Node(v);
24. tail = head;
25. return;
26. }
27. tail - > next = new Node(v);
28. tail = tail - > next;
29. }
30. void insert\_at\_head(int v) {
31. if (head == NULL) {
32. head = tail = new Node(v);
33. return;
34. }
35. Node \* new\_head = new Node(v);
36. new\_head - > next = head;
37. head = new\_head;
38. }
39. void insert\_at(int i, int v) {
40. //insert at index i, or at the end if greater than length
41. if (i == 0) {
42. insert\_at\_head(v);
43. return;
44. }
45. Node \* p = head;
46. while (p != NULL && --i)
47. p = p - > next;
48. if (i)
49. insert\_at\_tail(v);
50. else {
51. Node \* new\_node = new Node(v);
52. new\_node - > next = p - > next;
53. p - > next = new\_node;
54. }
55. }
56. void print() {
57. Node \* p = head;
58. if (p == NULL) return;
59. while (p != NULL) {
60. cout << p - > data << " ";
61. p = p - > next;
62. }
63. cout << endl;
64. }
65. };
66. int main() {
67. LinkedList l;
68. l.insert\_at\_head(1);
69. l.print();
70. l.insert\_at\_head(2);
71. l.print();
72. l.insert\_at\_tail(3);
73. l.insert\_at\_tail(4);
74. l.print();
75. l.insert\_at(2, 5);
76. l.print();
77. return 0;
78. }

**Output:**