```
#include <iostream>
#include <string>
#include <ctime>
#include <cstdlib>
using namespace std;

struct node {
  int data;
  int prev;
  int next;

  // Constructor:
  node() {
    data = -1;
    prev = -1;
    next = -1;
  }
};

node A[100];    // Array of nodes used to simulate the linked list
int temp = 1;   // Array index of the next free space (beginning from index 1 initially)
int head = -1;  // Array index of the starting element in the list (-1 implies no entries yet)
int last = -1;  // Array index of the last in the list
int elements = 0;   // Number of elements in the list

// Function prototypes:
void insert(int);
void insertAfter(int,int);
```

```
void deleteElement(int);
void reverse();
void show();
int findIndex(int);
```

```
void insert(int value) {
   if (temp == -1) {
      cout << "\nNo free space available.\n\n";
   } else {
      elements++;
      A[temp].data = value;
      A[temp].next = 0;
      if (head == -1) {
         // If the list is empty prior to the insertion
         A[temp].prev = 0;
         head = temp;
         last = temp;
      } else {
         A[last].next = temp;
         A[temp].prev = last;
         last = temp;
      }
      cout << endl << "Element \'" << value << "\'"<< " successfully inserted\n\n";
```

```
        // Finding the index of the next free location:
        do {
           do {
              temp = (temp + 1) % 100;
           } while(temp == 0);
           if (A[temp].next == -1) return;
        } while(temp != last);
        temp = -1;
        cout << "\nNo more free space available. Please delete some elements before inserting new.\n\n";
     }
}



void insertAfter(int value1,int value2) {
     if (temp == -1) {
        cout << "\nNo free space available.\n\n";
     } else {
        int pre = findIndex(value1);

        if (pre == -1) {
           cout << "\nElement " << value1 << " not found\n\n";
           return;
        }

        if (A[pre].next == 0) {
           // If value1 is the last element in the list
           insert(value2);
           return;
        }
```

```
        elements++;
        A[temp].data = value2;
        A[temp].prev = pre;
        A[temp].next = A[pre].next;
        A[A[pre].next].prev = temp;
        A[pre].next = temp;
        cout << endl << "Element \'" << value2 << "\'"<< " successfully inserted\n\n";

        // Finding the index of the next free location:
        int temp = temp;
        do {
           do {
              temp = (temp + 1) % 100;
           } while(temp == 0);
           if (A[temp].next == -1) return;
        } while(temp != temp);
        temp = -1;
        cout << "\nNo more free space available. Please delete some elements before inserting new.\n\n";
     }
}


void deleteElement(int value) {
    if (head == -1) {
        cout << "\nNo elements to delete.\n\n";
        return;
    }
```

```
if (elements == 1) {
   // Deleting the only element in the list
   A[head].data = -1;
   A[head].prev = -1;
   A[head].next = -1;
   head = -1;
   elements--;
   return;
}

int index = findIndex(value);

if (index == -1) {
   cout << "\nElement not found.\n\n";
   return;
}

if (index == head) {
   // Deleting the first element in the list
   int suc = findIndex(A[A[index].next].data);
   A[suc].prev = 0;
   head = suc;
   A[index].data = -1;
   A[index].prev = -1;
   A[index].next = -1;
   cout << endl << "Element \'" << value << "\'"<< " successfully deleted\n\n";
   elements--;
   return;
}
```

```
    if (index == temp) {
        // Deleting the last element in the list
        int pre = findIndex(A[A[index].prev].data);
        A[pre].next = 0;
        A[index].data = -1;
        A[index].prev = -1;
        A[index].next = -1;
        cout << endl << "Element \'" << value << "\'"<< " successfully deleted\n\n";
        elements--;
        return;
    }

    int suc   = A[index].next;
    int pre = A[index].prev;
    A[pre].next = suc;
    A[suc].prev = pre;
    A[index].data = -1;
    A[index].prev = -1;
    A[index].next = -1;
    cout << endl << "Element \'" << value << "\'"<< " successfully deleted\n\n";
    elements--;
}

void reverse() {
    int i = head;
    last = head;
    int temp;
    int hold;
```

```
    while (i != 0) {
        hold = i;
        i = A[i].next;

        //swap the previous and next data members to reverse the list:
        temp = A[hold].prev;
        A[hold].prev = A[hold].next;
        A[hold].next = temp;
    }
    head = hold;
}



void show() {
    cout << "\nThe current list is: ";
    int i = head;
    while (i != 0) {
        cout << A[i].data << " ";
        i = A[i].next;
    }
    cout << endl << endl;
}



int findIndex (int x) {
    // A helper function to return the index of the element x in A
    int i = head;
    while (i != 0) {
        if (A[i].data == x) {
```

```
            return i;
        }
        i = A[i].next;
    }
    return (-1);
}

void counter()
{

    int i = head;
    int h = 0;
    while (i != 0)
{
        if (A[i].data != NULL)
            {


            h++ ;
        }
        i = A[i].next;
    }
    cout<<"Number of values"<<h+1<<endl;;
}

int main() {
    int a, b;
    char c;
    cout << "Enter 'E' or 'e' to exit: \n";
    cout << "Enter 'I' or 'i' to insert element/data followed by the desired element/data: \n";
```

```
   cout << "Enter 'A' or 'a' to insert another element followed by the previous element/data to the list
\n";
   cout << "Enter 'D' or 'd' to delete element from the list: \n";
   cout << "Enter 'C' or 'c' to count elements : \n";
   cout << "Enter 'P' or 'p' to show the list: \n";




   do {
      cin >> c;
      switch(c) {
            case 'I':   srand((unsigned)time(0));
                  for(int i=0;i<100;i++)
                              {
                               a = (rand()%1000) + 1;
                  insert(a);
                                    }
                               break;
                   case 'i':   srand((unsigned)time(0));
                  for(int i=0;i<100;i++)
                             {
                               a = (rand()%1000) + 1;
                  insert(a);
                                    }


                  insert(a);
                  break;
            case 'A':   cin >> a >> b; //Enter the element after which you want to enter the new element
                  insertAfter(a,b);
```

```
                break;
        case 'a':   cin >> a >> b; //Enter the element after which you want to enter the new element
                insertAfter(a,b);
                break;
        case 'D':   cin >> a;
                deleteElement(a);
                break;
        case 'd':   cin >> a;
                deleteElement(a);
                break;
        case 'P':   show();
                break;
        case 'p':   show();
                break;
        case 'E':   cout << "\nExiting program...\n\n";
                break;
        case 'e':   cout << "\nExiting program...\n\n";
                break;
                 case 'C':   cout<<"\n Counting the elements:\n";
                counter();
                                break;
                 case 'c':   cout<<"\n Counting the elements:\n";
                counter();
                                break;
        default:    cout << "\nInvalid input entered\n\n";
   }

} while (c != 'E');
```

```
    return 0;
}
```

SCREENSHOT OF OUPTUT