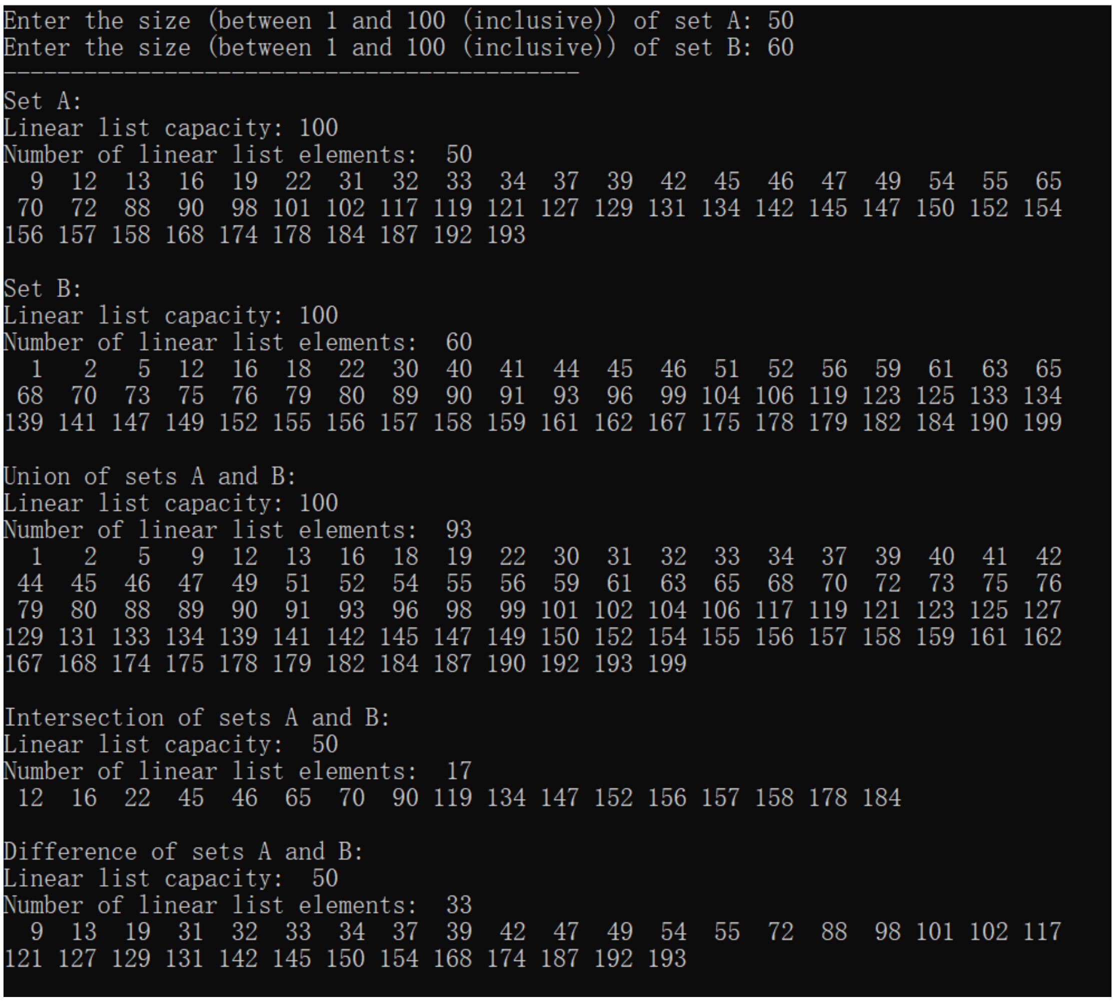
Programming Practice: Linear Lists

1. Assuming that the ordered linear list is **non-duplicate**, **ascending order**, and its elements are random integers between 0 and 200, the memory capacity of this linear table will be dynamically adjusted. The constant SEGMENT is the length of each segment of the ordered linear list elements. When the linear table is initialized, the memory capacity of the linear list is set to one segment of elements, that is, the initial value of capacity is SEGMENT; whenever an element is inserted, if the linear list is full, the memory capacity is increased by a segment of elements; whenever an element is deleted , if the memory capacity of the linear list exceeds the number of elements by two segments, the memory capacity will be reduced by one segment.

   Write a program in the C programming language to implement a set as a non-duplicate ordered linear list with dynamic arrays. In the main program, create two sets A and B (assuming that the number of elements of a set does not exceed 100), and print the two sets and compute: (1) union of A and B; (2) intersection of A and B; (3) difference set of A and B. Print the result of each operation.

   Program solution: linear_list_dynamic_set.rar. Example of program execution:



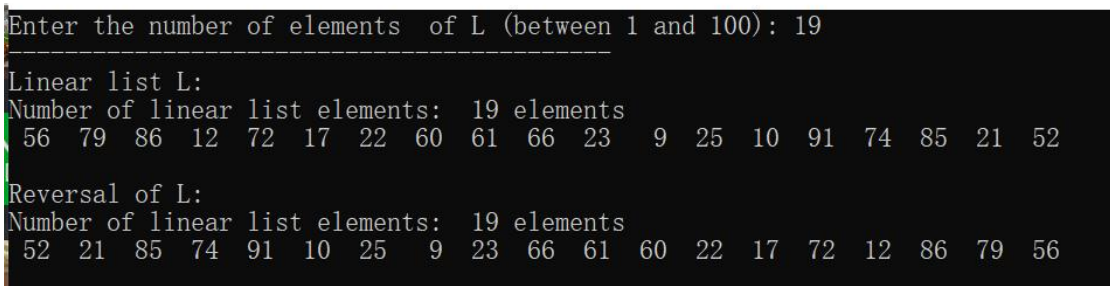
```
Enter the size (between 1 and 100 (inclusive)) of set A: 50
Enter the size (between 1 and 100 (inclusive)) of set B: 60
-------------------------------------------
Set A:
Linear list capacity: 100
Number of linear list elements:  50
  9  12  13  16  19  22  31  32  33  34  37  39  42  45  46  47  49  54  55  65
 70  72  88  90  98 101 102 117 119 121 127 129 131 134 142 145 147 150 152 154
156 157 158 168 174 178 184 187 192 193

Set B:
Linear list capacity: 100
Number of linear list elements:  60
  1   2   5  12  16  18  22  30  40  41  44  45  46  51  52  56  59  61  63  65
 68  70  73  75  76  79  80  89  90  91  93  96  99 104 106 119 123 125 133 134
139 141 147 149 152 155 156 157 158 159 161 162 167 175 178 179 182 184 190 199

Union of sets A and B:
Linear list capacity: 100
Number of linear list elements:  93
  1   2   5   9  12  13  16  18  19  22  30  31  32  33  34  37  39  40  41  42
 44  45  46  47  49  51  52  54  55  56  59  61  63  65  68  70  72  73  75  76
 79  80  88  89  90  91  93  96  98  99 101 102 104 106 117 119 121 123 125 127
129 131 133 134 139 141 142 145 147 149 150 152 154 155 156 157 158 159 161 162
167 168 174 175 178 179 182 184 187 190 192 193 199

Intersection of sets A and B:
Linear list capacity:  50
Number of linear list elements:  17
 12  16  22  45  46  65  70  90 119 134 147 152 156 157 158 178 184

Difference of sets A and B:
Linear list capacity:  50
Number of linear list elements:  33
  9  13  19  31  32  33  34  37  39  42  47  49  54  55  72  88  98 101 102 117
121 127 129 131 142 145 150 154 168 174 187 192 193
```

2. Repeat Question 1 using single linked order linear list. (No solution provided.)

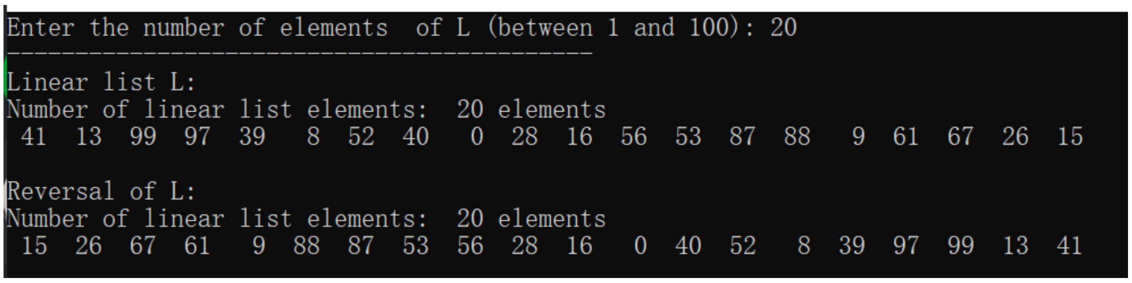
3. Write a Dev-C++ project to define and implement a non-ordered circular doubly linked list DList with no duplicated elements. Function DiLst reversal(DList) is

the reversal operation of the circular doubly linked list, that is, the order of a circular doubly linked list is reversed. Insert a new node to the tail of the linear list. Swap data elements to perform reversal operation. Generate a circular doubly-linked list L of 1 to 100 elements with values between 0 and 999, print L and reversal(L).

Program solution: circular_double_list_reversal_swap_data.rar. Example of program execution:



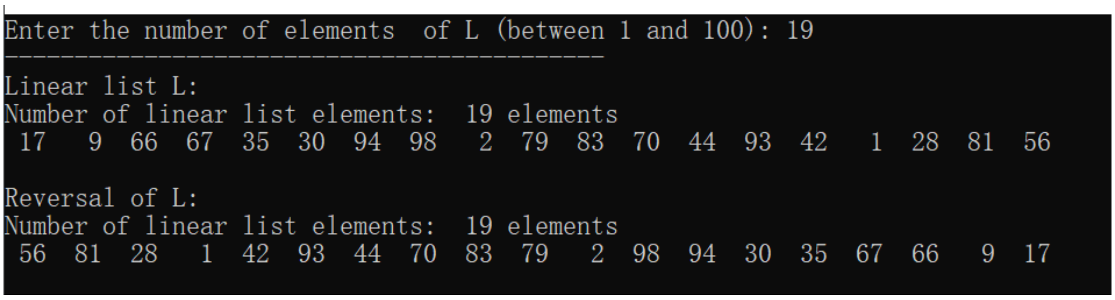
```
Enter the number of elements  of L (between 1 and 100): 19
-------------------------------------------
Linear list L:
Number of linear list elements:  19 elements
 56  79  86  12  72  17  22  60  61  66  23   9  25  10  91  74  85  21  52

Reversal of L:
Number of linear list elements:  19 elements
 52  21  85  74  91  10  25   9  23  66  61  60  22  17  72  12  86  79  56
```



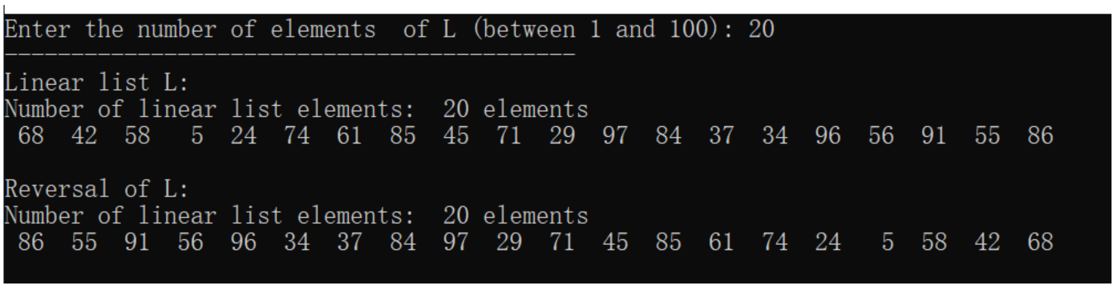
```
Enter the number of elements  of L (between 1 and 100): 20
-------------------------------------------
Linear list L:
Number of linear list elements:  20 elements
 41  13  99  97  39   8  52  40   0  28  16  56  53  87  88   9  61  67  26  15

Reversal of L:
Number of linear list elements:  20 elements
 15  26  67  61   9  88  87  53  56  28  16   0  40  52   8  39  97  99  13  41
```

4. Repeat Question 3 by physically swap two corresponding nodes, i.e., manipulate the links of two swapping nodes.

Program solution: circular_double_list_reversal_swap_node.rar. Example of program execution:



```
Enter the number of elements  of L (between 1 and 100): 19
-------------------------------------------
Linear list L:
Number of linear list elements:  19 elements
 17   9  66  67  35  30  94  98   2  79  83  70  44  93  42   1  28  81  56

Reversal of L:
Number of linear list elements:  19 elements
 56  81  28   1  42  93  44  70  83  79   2  98  94  30  35  67  66   9  17
```

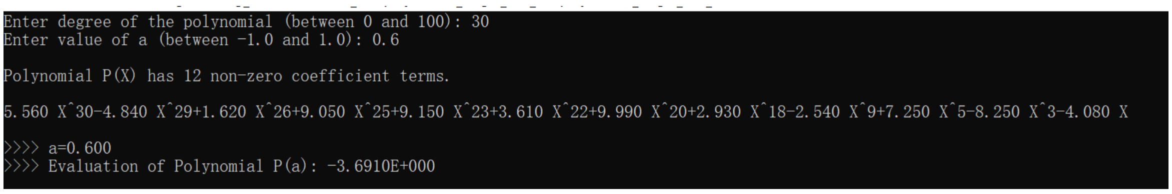


```
Enter the number of elements  of L (between 1 and 100): 20
-------------------------------------------
Linear list L:
Number of linear list elements:  20 elements
 68  42  58   5  24  74  61  85  45  71  29  97  84  37  34  96  56  91  55  86

Reversal of L:
Number of linear list elements:  20 elements
 86  55  91  56  96  34  37  84  97  29  71  45  85  61  74  24   5  58  42  68
```

5. A polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$, where $-10<a_i<10$ and $a_n \neq 0$, is represented using an ordered single link linear list. Suppose the highest term and coefficients are generated using random numbers; and there is only 33% chance that the coefficients are not 0 except for the coefficient of the highest degree term.

Write a C project to generate polynomial p(x) and evaluate p(a) for $-1.0<a<1.0$.

Program solution: polynomial_single_link_list.rar. Example of program execution:



```
Enter degree of the polynomial (between 0 and 100): 30
Enter value of a (between -1.0 and 1.0): 0.6

Polynomial P(X) has 12 non-zero coefficient terms.

5.560 X^30-4.840 X^29+1.620 X^26+9.050 X^25+9.150 X^23+3.610 X^22+9.990 X^20+2.930 X^18-2.540 X^9+7.250 X^5-8.250 X^3-4.080 X

>>>> a=0.600
>>>> Evaluation of Polynomial P(a): -3.6910E+000
```