Problem 1:
Polynomial Problem
Problem 1: Polynomial Problem

- Use dynamic arrays to implement a polynomial class

<table>
<thead>
<tr>
<th>Class Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Constructors:</strong></td>
</tr>
<tr>
<td>- Polynomial() as default constructor;</td>
</tr>
<tr>
<td>- <strong>Private members:</strong></td>
</tr>
<tr>
<td>- Index coefficient - integer value;</td>
</tr>
<tr>
<td>- X value - pointer integer value;</td>
</tr>
<tr>
<td>- <strong>Public members:</strong></td>
</tr>
<tr>
<td>- Get/Set index coefficient;</td>
</tr>
<tr>
<td>- Get/Set index value;</td>
</tr>
</tbody>
</table>

main() must have the variable that stores the value of X
Problem 1: Polynomial Problem

- We read a string with the values from the user:

\[2\ 7\ 5\ 4\ 2\ 9\ 0\]

First value indicates the \(X\) value of the polynomial.
Problem 1: Polynomial Problem

- We read a string with the values from the user:

```
2 7 5 4 2 9 0
```

Other values are the coefficient positions.
Problem 1: Polynomial Problem

- We read a string with the values from the user: 2 7 5 4 2 9 0

Make sure that the same coefficient cannot be added twice

Other values are the coefficient positions
Problem 1: Polynomial Problem

- We read a string with the values from the user:

  2 7 5 4 2 9 0

  Other values are the coefficient positions

  Make sure that the same coefficient cannot be added twice

  If a alphabetic string is used, then the program should return 0, as shown in the Quiz output example.
Problem 1: Polynomial Problem

- We read a string with the values from the user:

\[ f(x) = x^9 + x^7 + x^5 + x^4 + x^2 + x^0 \]
Problem 1: Polynomial Problem

- We read a string with the values from the user:

\[
2 7 5 4 2 9 0
\]

\[
f(2) = 2^9 + 2^7 + 2^5 + 2^4 + 2^2 + 2^0
\]
Problem 1: Suggestions

1. Use `getline(cin, input)` to read the string from user

2. To get the individual values from the read input we can use `stringstream`:

   ```
   string input;
   getline(cin, input);
   stringstream stream(input);
   while(1) {
       int n;
       stream >> n;
       if(!stream) break;
       // … do whatever you want with ‘n’
   }
   ```

3. You could use vectors to store the read value temporarily in this example
Problem 1: Suggestions

4. After you have read the values, you must identify and remove the numbers that are repeated;

a. If you used vector to store numbers temporarily, then:

```
int size = vec.size();
for (int i ... ) {
    for (int j ... ) {
        if (vec[j] is equal to vec[i])
            vec.erase(vec.begin() + j);
        if(size != vec.size()) {
            --j; size = vec.size();
            /* because we removed an entry, we need to update index */
        }
    }
}
```
Problem 1: Suggestions

4. After you have read the values, you must identify and remove the numbers that are repeated;

   a. If you used vector to store numbers temporarily, then:

   ```
   int size = vec.size();
   for (int i ... ) {
       for (int j ... ) {
           if (vec[j] is equal to vec[i])
               vec.erase(vec.begin() + j);
           if(size != vec.size()) {
               --j; size = vec.size();
               /* because we removed an entry, we need to update index */
           }
       }
   }
   ```

   It erases the repeated number

5. Then you can create the dynamic array that stores the necessary number of polynomial;
Problem 2: Polynomial Using Linked list
Problem 2: What are Linked Lists?

- Linked lists are one of the most famous data structures.
Let’s develop our number list using linked lists;

structure Polynomial
- Pointer to the X value;
- Integer to store the coefficient;
- Pointer to next structure Todo;

Class List()
- Constructors:
  - List() as default constructor;
- Private members:
  - Pointer to head and tail struct Polynomial;
- Public members:
  - Add an polynomial to the list;
  - Return the head pointer to the list;
Problem 2: Suggestions

Empty list!
Problem 2: Suggestions

- Dynamically allocate and initialize the structure

Adding an entry to the list!
Problem 2: Suggestions

Adding an entry to the list!
Problem 2: Suggestions

Adding an entry to the list!

Dynamically allocate and initialize the structure
Problem 2: Suggestions

Adding an entry to the list!

Dynamically allocate and initialize the structure
Problem 2: Suggestions

Adding an entry to the list!

Dynamically allocate and initialize the structure
Problem 2: Suggestions

Get the pointer to the head node

Iterating in the list!

Head -> Tail

Data - next

Data - next

NULL
Problem 2: Suggestions

Now we are here, to go to the next, we access the next pointer.

Iterating in the list!
Problem 2: Suggestions

Iterating in the list!

We are in the last, we need to have a stop condition! How?