DISCUSSION

When using Dynamic Arrays,

Advantages:

- Easier to manage than linked lists because their links are always there (only lose control when having no pointer to store the address or not knowing their size).

Disadvantages:

- Must use many functions to work with them.
- Must use additional variables to store their lengths.

When using Linked Lists,

Advantages:

- Save more memory spaces because of only allocating spaces to non-zero terms.
- Easier to maintain and expand the code since using classes to implementing them.

Disadvantages:

- Higher risk of memory leaks as their links are more breakable (a mistake in executing any actions on them may cause leaks).
- Must check whether they are empty or not before doing anything on them.

If the program using static arrays instead of dynamic arrays or linked lists, there are some obvious assumptions we can make.

- The program can only handle a certain degree polynomial. For example, if the arrays used are at 100 elements long, they can only handle polynomials with the degree of 99.
- There will be a large number of unused spaces when the input polynomials' degrees are much lower than the maximum one the code is designed to take.
- However, the program will be much straightforward to implement because we do not need to manually delete memory spaces allocated on the heap at the end of the program.

To enhance the error checking, I suggest listing all the legal characters of the polynomials. After that, we use a loop ("the loop") to inspect each character of the input. If the current character is legal, we will look at the next character(s) to see if it is also legal. If it is, we continue the loops. Later on, we could think about adding more types of legal characters or phrases such as parentheses. When adding more legal characters, we must decide which characters they can go after and which ones can go after them and implement those things to the loop.

If float values can take places in the coefficients of the input polynomials, we will add decimal point "." to the list of legal characters. "." can only go after a number character, and only another number character can go after it. After the error checking (which initially I allow "." to appear in the input), we will remove the restriction of float values while reading and extracting coefficients and powers of the input.

Known Bugs:

1. The program cannot distinguish the difference in the case of "1.0" and "1.0.0" and many other similar ones when it comes to the stage of error checking.