**CSCI6033\_Lab2\_Yibo\_Zhu**

**Constructor-Array (int size, string input)**

Precondition:

Size is a “int” type, bigger than 0

Input is a string of comma-separated characters

Postcondition:

Input string is stored in a char array in order.

A close up of a sign

Description automatically generated



CWE-20: Improper Input Validation



A screenshot of a cell phone

Description automatically generated



I use improper input size = -10 to test the precondition. To handle this error, I use a while loop to request a new input for size, if the size still is less than 0, repeat the request.

**Destructor- ~Array()**

Precondition:

Char\* array = new char[size] (new memory is allocated).

Postcondition:

Deallocate the memory.

**Char ReadFromArray (int index)**

Precondition:

Index is valid (0 <= index < size)

Postcondition:

Return the character at the specific index

A screenshot of a cell phone

Description automatically generated



A picture containing table

Description automatically generated



CWE-125: Out-of-bounds Read

I use out-of-bound index and try to read the array use that index. To handle the error, I use a while loop(index <0 ||index >= size), if index is not fit the condition, request for a new input.

A screenshot of a cell phone

Description automatically generated



**Void WriteToArray (int index, char replace)**

Precondition:

Index is valid (0 <= index < size)

Char replace is valid.

Postcondition:

The character at specific index is replaced by char replace.

A screenshot of a cell phone

Description automatically generated



CWE-125: Out-of-bounds Read

I use out-of-bound index and try to overwrite a new character into the array. To handle the error, I use a while loop (index <0 ||index >= size), if index is not fit the condition, request for a new input.

A screenshot of a cell phone

Description automatically generated



**Void DeleteArray()**

Precondition:

Char\* array = new char[size] (new memory is allocated).

Postcondition:

Deallocate the memory.

Set the array to NULL.

A screenshot of a cell phone

Description automatically generated



Repeat delete behavior won’t trigger an error, but it is an invalid behavior. I test it by selecting choice 3 repeatedly. To handle the invalid behavior, I use an if-else statement, if char\* array = NULL, show the alert info then stop the behavior.

A screenshot of a cell phone

Description automatically generated



**Void PrintArray()**

Precondition:

Array is created and has characters stored in it.

Postcondition:

Display the content of the array.

A screenshot of a cell phone

Description automatically generated



CWE-416: Use After Free

I delete the array first then try to print the array. To handle the error, I use a if-else statement, if char\* array == NULL, redirect to the menu and suggest the user select choice 5.

A screenshot of a cell phone

Description automatically generated



**Void NewArray (int size, string input)**

Precondition:

Size is a “int” type, bigger than 0

Input is a string of comma-separated characters

Postcondition:

Input string is stored in a char array in order.

A screenshot of a cell phone

Description automatically generated



CWE-20: Improper Input Validation

I use improper input size = -10 to test the precondition. To handle this error, I use a while loop to request a new input for size, if the size still is less than 0, repeat it.

A screenshot of a cell phone

Description automatically generated



**Graduate students question:**

Yes, python assert keywork helps in smooth flow of code. Like C++, it takes a Boolean condition, which when returns true keep running the program, but if it is computed to be false, it raises an AssertionError along with the optional message provided.

Example:

A screenshot of a cell phone

Description automatically generated

A close up of a sign

Description automatically generated

Reference:

<https://www.geeksforgeeks.org/python-assert-keyword/>