ET-580 - Pointers & Dynamic Arrays - Homework

Reading

Chapter 10.3 Classes, Pointers and Dynamic Arrays

Implementation

1. Implement the following using a **Dynamic Array** and **Pointer Arithmetic**:

a) Use the provided main function (see below).
b) ***Populate*** function accepts an array and its size by parameter, then stores values from 0 to size-1 using pointer arithmetic.
c) ***Print*** function accepts an array and its size by parameter, then prints the array using pointer arithmetic.
d) ***PrintMemory*** accepts an array and its size by parameter, then prints array memory values using pointer arithmetic.
e) ***Grow*** function accepts an array, its size and a new size by parameter, then returns a new dynamic array which is a larger copy of the original.

Use the *printMemory* function to verify that the old and new array have unique separate memory spaces as in the output example, otherwise the code is wrong.

Main

```
int main( ) {
    cout << endl;

    int size, newSize;
    cout << "Enter a size: ";
    cin >> size;
    cout << "\n";

    int *p = new int[size]();
    cout << "Original: " << "\n";
    populate(p, size);
    print(p, size);
    printMemory(p, size);

    cout << "Enter a new size: ";
    cin >> newSize;

    p = grow(p, size, newSize);
    cout << "After grow: " << "\n";
    print(p, newSize);
    printMemory(p, newSize);

    cout << endl;
    return 0;
}
```

Correct Output

```
Enter a size: 2

Original:
0 1
0x7fe00a4057d0
0x7fe00a4057d4

Enter a new size: 4

Inside grow:
0 1 0 0
0x7fe00a4057e0
0x7fe00a4057e4
0x7fe00a4057e8
0x7fe00a4057ec

After grow:
0 1 0 0
0x7fe00a4057e0
0x7fe00a4057e4
0x7fe00a4057e8
0x7fe00a4057ec
```

2. Use Dynamic programming to implement a **memoized** solution for the classic Fibonacci Sequence using heap memory for storage.

   a. Compare this solutions performance to the classic recursive algorithm for values of n up to 46.

   b. Draw and compare the recursion tree for the memoized algorithm.

   c. What is the time complexity and how does it compare to the original?

3. Use Dynamic programming to implement a **tabulated** solution for the classic Fibonacci Sequence using heap memory for storage.

   a. Compare this solutions performance to the classic recursive algorithm for values of n up to 46.

   b. What is the time complexity and how does it compare to the original?