Due date: Mar 31, 11:59 PM

Homework 3

**1. Implement and grow a dynamic array using pointer arithmetic.**

a) Use the provided main function (see below).

b) Implement a **populate** function which stores values from 0 to size into the array p using pointer arithmetic to access array locations.

c) Implement a **print** function which prints the values of the array p using pointer arithmetic.

d) Implement a **printMemory** function which prints the memory addresses of all elements in array p using pointer arithmetic.

e) Implement a **grow** function which resizes the existing array from the initial size to a new size using pointer arithmetic.

f) Verify via the output that the new array is a distinct memory space from the original array.

Main:

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

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

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

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

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

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

Output Example:

```
Enter a size: 5

Original:
0 1 2 3 4
0x7f970bd04080
0x7f970bd04084
0x7f970bd04088
0x7f970bd0408c
0x7f970bd04090


Enter a new size: 3

Inside grow:
0 1 2
0x7f970bd040a0
0x7f970bd040a4
0x7f970bd040a8


After grow:
0 1 2
0x7f970bd040a0
0x7f970bd040a4
0x7f970bd040a8
```

Due date: Mar 31, 11:59 PM

Use following main function to test your program.

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

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

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

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

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

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

**2. Pointer arithmetic. Implement the following:**

a. Implement a print function with array and size parameters. This function should print the array using pointer arithmetic.

b. Overload the print function to accept array, size and index parameters. If the index is between 1 and size-2 inclusive:

Use pointer arithmetic to print:
1. the value at index
2. the value previous to the index
3. the value after the index

c. Implement a sum function with array and size parameters. This function should return sum of the array, and use pointer arithmetic.

Due date: Mar 31, 11:59 PM

d. Implement a average function with array and size parameters. This function should return average of the array, and use pointer arithmetic.

e. Use provided main function to test your functions.

Main function:

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

    int *a = new int[5] {5,15,25,35,45};
    print(a, 5);
    print(a, 5, 2);
    cout<<endl<<sum(a,5)<<endl;
    cout<<endl<<average(a,5)<<endl;

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

Output Example

```
5 15 25 35 45
15 25 35
125

25
```

**Grading policy**:

Should submit .cpp file format, other format will be not accepted and assigned 0 point directly.

50% points loss if the program doesn't compile. 50% points for the rest.
If the code compiles and runs, full points if it succeeds for all requirements.