**CSCI 1730 - Programming Assignment 3 - 100 pts.**

**Due Date: Apr 3, 2019**

1. Create a **class**, called Complex, for modeling complex numbers, *a* + *bi*, and some complex number arithmetic/comparison operations.

Here is what should be included within this class:

* Include a no-argument constructor (to initialize a complex number to 0+0*i*).
* Include public member functions to perform these complex number tasks:
  + Addition of complex numbers
  + Subtraction of complex numbers
  + Multiplication of complex numbers
  + Division of complex numbers
  + User input of a complex number
  + Display of a complex number
  + Conversion of a real number, *r*, to a complex number, *r* + 0*i*
  + Check for equality of two complex numbers

Then, write a **C++ program** that will use the Complex class to repeatedly do one of the following tasks:

1. *Perform a complex number arithmetic operation.* For this option, the program will ask the user to enter a complex number, an arithmetic operation (+, –, \*, /), and a second complex number, and will then calculate and display the result of performing the arithmetic operation on the two input complex numbers.
2. *Determine if a complex number is a solution of a quadratic equation.* For this option, the program will prompt for and read in the real number coefficients, *a*, *b*, and *c*, of a quadratic equation, *ax*2+*bx*+*c=0*. Next, it will prompt for and read in a complex number, *z*. Then, it will determine if *z* is a solution of the quadratic equation.

* Note: When checking for equality of two complex numbers, do not use the “is equal to” operator on the float values - instead, determine if the absolute value of the float values are smaller than a threshold value (something small, like 0.000001).
* To extract input of a complex number a + bi from keyboard, do the following:

double a, b;

char i;

cin >>a >> b >> i;

Complex Number Review:

A complex number is a number of the form *a* + *bi* where *a* and *b* are real numbers and *i* is the imaginary unit, .

* Addition/Subtraction: 
* Multiplication: 
* Division: 

Here is output from a sample run of the program (user input in **bold**):

Select an option - (1) perform complex number arithmetic

(2) check for quadratic equation solution

(3) exit

**1**

Enter a complex number a+bi: **2+3i**

Enter an operation (+, -, \*, /): **+**

Enter a complex number a+bi: **4-8i**

2+3i + 4-8i = 6-5i

Select an option - (1) perform complex number arithmetic

(2) check for quadratic equation solution

(3) exit

**1**

Enter a complex number a+bi: **2+9i**

Enter an operation (+, -, \*, /): **-**

Enter a complex number a+bi: **4+5i**

2+9i - 4+5i = -2+4i

Select an option - (1) perform complex number arithmetic

(2) check for quadratic equation solution

(3) exit

**1**

Enter a complex number a+bi: **4+2i**

Enter an operation (+, -, \*, /): **\***

Enter a complex number a+bi: **4-2i**

4+2i \* 4-2i = 20+0i

Select an option - (1) perform complex number arithmetic

(2) check for quadratic equation solution

(3) exit

**1**

Enter a complex number a+bi: **4+8i**

Enter an operation (+, -, \*, /): **/**

Enter a complex number a+bi: **1-1i**

4+8i / 1-1i = -2+6i

Select an option - (1) perform complex number arithmetic

(2) check for quadratic equation solution

(3) exit

**2**

Enter the coefficients of a quadratic equation: **1 -2 5**

Enter a complex number a+bi: **1+2i**

The complex number: 1+2i is a solution of the quadratic equation

Select an option - (1) perform complex number arithmetic

(2) check for quadratic equation solution

(3) exit

**2**

Enter the coefficients of a quadratic equation: **1 -2 5**

Enter a complex number a+bi: **2+3i**

The complex number: 2+3i is not a solution of the quadratic equation

Select an option - (1) perform complex number arithmetic

(2) check for quadratic equation solution

(3) exit

**3**

2. Modify the Complex **class** from problem #1 and replace all arithmetic and relational operator member functions with appropriate overloaded operators (+, -, \*, /, ==). In addition, add **friend** functions to overload the insertion and extraction operators (<< and >>) for use with the Complex **class**. Then modify the main function and any other stand-alone functions that make use of Complex objects to make use of the new overloaded operators.

3. Write a **class** IntSet for modeling sets of integers in the range 0 through 99. A set should be represented internally as an array of type bool: The ith array element will be true whenever integer i is in the set and will be false whenever integer iis not in the set. Include a no-argument constructor that initializes a set to the so-called “empty set,” i.e., a set whose array representation contains all false values. The class should include the following overloaded operators:

+ to perform the union of two set (the union of sets A and B is the set that contains all

elements of set A or set B, or both).

\* to perform the intersection of two sets (the intersection of sets A and B is the set that

contains all elements in *both* set A and set B.)

- to form the difference of two sets (the difference of sets A and B is the set containing those elements that are in A but not in B)

+= to add an integer into a set.

-= to delete an integer from a set.

== to determine if two sets are equal.

! to form the complement of a set (the complement of set *A*, denoted , is the set containing all the elements in the universal set that are not in A - the universal set for this problem is the set containing all integers between 0 and 99)

<= to determine if one set is a subset of another set (set *A* is a subset of set *B* means that for any element *x* in *A*, *x* is also an element of set *B*).

<< to display a set in roster notation (for example, {2, 3, 5, 9})

**Requirement:** The overloaded += and -= operators should check for valid integer input (in the range 0-99), or if an add-item is already in the set, or if a delete-item is not in the set. An error message for invalid input should be generated.

Then, write a **C++ program** that uses the new IntSet class. The program should allow the user to repeatedly select from these options:

* add numbers to a set
* remove numbers from a set
* form the union of two sets
* form the intersection of two sets
* form the difference of two sets
* determine if two sets are equal
* form the complement of an set
* determine if one set is a subset of another set
* display a set

The program should allow for up to ten sets to be created during a given program run. Use any stand-alone functions you feel necessary.

Here is output from a sample run of the program (user input in **bold**):

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**1**

Add numbers to which sets (A, B,C, D, E, F, G, H, I, J)? :**A**

Enter number to add: **1**

Add another (y or n): **y**

Enter number to add: **3**

Add another (y or n): y

Enter number to add**: 8**

Add another (y or n): n

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**9**

Display set (A, B, C, D, E, F, G, H, I, J)? :**A**

{1, 3, 8}

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**1**

Add numbers to which sets (A, B,C, D, E, F, G, H, I, J)? :**B**

Enter number to add: **2**

Add another (y or n): **y**

Enter number to add: **3**

Add another (y or n): y

Enter number to add: **7**

Add another (y or n): n

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**2**

Remove numbers from which set (A, B, C, D, E, F, G, H, I, J)? :**A**

Enter number to remove: **1**

Remove another (y or n): **n**

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**3**

Sets union – specify sets to use:

First set: (A, B, C, D, E, F, G, H, I, J)? :**A**

Second set: (A, B, C, D, E, F, G, H, I, J)? :**B**

Result set: (A, B, C, D, E, F, G, H, I, J)? :**C**

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**4**

Set intersection – specify sets to use:

First set: (A, B, C, D, E, F, G, H, I, J)? :**A**

Second set: (A, B, C, D, E, F, G, H, I, J)? :**B**

Result set: (A, B, C, D, E, F, G, H, I, J)? :**D**

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**5**

Set difference – specify sets to use:

First set: (A, B, C, D, E, F, G, H, I, J)? :**A**

Second set: (A, B, C, D, E, F, G, H, I, J)? :**B**

Result set: (A, B, C, D, E, F, G, H, I, J)? :**E**

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**6**

Set equality – specify sets to compare:

First set: (A, B, C, D, E, F, G, H, I, J)? :**A**

Second set: (A, B, C, D, E, F, G, H, I, J)? :**B**

These sets are not equal

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**7**

Set complement – specify sets to use:

Complement set: (A, B, C, D, E, F, G, H, I, J)? :**A**

Result set: (A, B, C, D, E, F, G, H, I, J)? :**F**

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**8**

Subsets – specify sets to compare:

First set: (A, B, C, D, E, F, G, H, I, J)? :**A**

Second set: (A, B, C, D, E, F, G, H, I, J)? :**B**

The first set is not a subset of the second

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**9**

Display set (A, B, C, D, E, F, G, H, I, J)? :**B**

{2, 3, 7}

Select an option:

1. - add numbers to a set
2. - remove numbers from a set
3. - form the union of two sets
4. - form the intersection of two sets
5. - form the difference of two sets
6. - determine if two sets are equal
7. - form the complement of an set
8. - determine if one set is a subset of another set
9. - display a set
10. – Exit

**10**

4. Write a **C++ program** that will read in one or more lines of text and then determine the following:

* The number of uppercase letters in the text.
* The number of lowercase letters in the text.
* The number of digits in the text.
* The number of whitespace characters (i.e., blanks or newlines) in the text.
* The number of other, non-letter, non-digit, non-whitespace characters in the text.

After reading the text, the program should display the text followed by a display of all of the above statistics.

**Suggestions and hints:**

* Use a string variable to read the text into (since you do not know in advance how much text will be input).
* The getline function can be used to read in more than one line into a string variable. To do this, specify a special terminating character as a “flag” to indicate input termination as the third parameter to getline when it is called to read. For example, getline(cin,text,'@'); will read text into the string variable text until a '@' character is encountered.
* You may assume that the text to be entered does not contain the '@' character.

Output from a sample run of the program (user input is one line in **bold**):

**A string is a joy forever! Enter 3 or 4 words: Am I a palindrome? Hello friends# Is this a valid**

**English sentence@**

A string is a joy forever! Enter 3 or 4 words: Am I a palindrome? Hello friends# Is this a valid

English sentence

There were 113 total characters.

There were 7 upper case letters.

There were 78 lower case letters.

There were 2 digits.

There were 22 white space characters.

There were 4 other characters.

5. Write a **C++ program** that repeatedly prompts for and reads an e-mail address and then determines and displays whether the address is valid or invalid. For purposes of this program, an e-mail address is valid if it satisfies these conditions:

* The address cannot contain blanks. For example, *john smith@minneapolis.edu* is not a valid e-mail address.
* The *@* character must occur exactly once. For example, *baz.cs.dpu.edu* and *bar@cs@dpu* are not valid e-mail addresses.
* The *@* character cannot be the first character of the address. For example, *@cs.dpu.edu* is not a valid e-mail address.
* Every occurrence of the dot character (.)must have a non-@, non-dot character on both side. For example, *bar@cs.* , *.ed@comcast.net*, and *bar@.depaul*, *joe..smith@bob.com* are not valid e-mail addresses.

After reading an email address, the program should display it. If the address is valid, a message should be displayed stating that it is. For invalid addresses, the program should generate an error message for each of the above conditions that was violated.

**Suggestions and hints:**

* Use string class strings in your program.
* Write a separate bool-valued function to check each of the four invalidity checks given above. Each should receive the email address via a parameter and then return true if the email address is invalid according to the particular invalidity conditions.
* Use a bool array of size four to store the results of calling each of these invalidity check functions.
* If you have buffering problems when repeatedly reading strings and chars, make use of the function call cin.ignore(80,’\n’) to clear the buffer at appropriate times.

Output from a sample run of the program (user input is one line in **bold**):

Enter the address: ***john smith@minneapolis.edu***

You entered: *john smith@minneapolis.edu*

Not valid – contains a blank

Enter another (y or n)? **y**

Enter the address: **baz.cs.dpu.edu**

You entered: baz.cs.dpu.edu

Not valid – not exactly one ‘@’

Enter another (y or n)? **y**

Enter the address: **bar@cs@dpu**

You entered:bar@cs@dpu

Not valid – not exactly one ‘@’

Enter another (y or n)? **y**

Enter the address: ***@cs.dpu.edu***

You entered: *@cs.dpu.edu*

Not valid – ‘@’ is first character

Enter another (y or n)? **y**

Enter the address: ***bar@cs.***

You entered: *bar@cs.*

Not valid – a dot is first or last, or preceded or followed by @ or .

Enter another (y or n)? **y**

Enter the address: ***.ed@comcast.net***

You entered: *.ed@comcast.net*

Not valid – a dot is first or last, or preceded or followed by @ or .

Enter another (y or n)? **y**

Enter the address: ***bar@.depaul***

You entered: *bar@.depaul*

Not valid - a dot is first or last, or preceded or followed by @ or .

Enter another (y or n)? **y**

Enter the address: ***joe..smith@bob.com***

You entered: *joe..smith@bob.com*

Not valid - a dot is first or last, or preceded or followed by @ or .

Enter another (y or n)? **n**

6. The following **C++** main driver, along with function myFunc, uses a **C++ class** DynArray that models a dynamic integer array – that is, the class uses dynamic memory allocation to create a contiguous block of memory for storing a specified number of integers. The indexing for a DynArray object is the same as for a regular array. But, a DynArray can be initialized to size zero.

Write the **C++** DynArray **class**. Here is a brief description of all of the class functions that your class should include:

* No-argument constructor – initializes a DynArray object to being empty.
* One-argument constructor – uses dynamic memory allocation to obtain a contiguous block of memory for storing *n* int values, where *n* is its argument.
* show – displays the *n*th element in the DynArray. If the DynArray is empty or if *n* is an invalid index, this function should generate an error message.
* set – will set the *n*th element in the DynArray to *x*, where *n* is the value of its first argument and *x* is value of its second argument. If the DynArray is empty or if *n* is an invalid index, this function should generate an error message.
* expand – will take an existing DynArray and expand its size by its argument, *s*. **Hint:** To expand a DynArray, allocate a new, larger block of dynamic memory, copy the values from the old DynArray to the new memory, and deallocate the old memory.
* A destructor to deallocate dynamic memory when a DynArray object passes out of scope.

**Requirement:** When accessing the dynamic array elements in the set, show and expand member functions, you **must use the dereferencing operator, \*,** along with pointer arithmetic instead of the array indexing operator, [].

Next, combine your DynArray class with the following main and myFunc code and run the resulting **C++ program**. The output generated from a run of your program should be similar to that shown in the output of a sample run given after the code.

void myFunc();

int main()

{

int size,more,i;

DynArray y;

cout << "Enter dynamic array size: ";

cin >> size;

DynArray x(size);

for(i=0;i<size;i++)

x.set(i,3\*i);

for(i=0;i<size;i++)

x.show(i);

cout << "How much more dynamic array space do you want? ";

cin >> more;

x.expand(more);

for(i=0;i<(size+more);i++)

x.set(i,5\*i);

for(i=0;i<(size+more);i++)

x.show(i);

x.show(size+more+5); //invalid index in show

x.set(-2,9); //invalid index in set

y.set(3,6); //empty DynArray set

y.show(3); //empty DynArray show

myFunc();

char ch; cin >> ch;

return 0;

}

void myFunc()

{

int i;

cout << "hi from myFunc...\n";

DynArray y(5);

for(i=0;i<5;i++)

y.set(i,i\*i);

for(i=0;i<5;i++)

y.show(i);

cout << "bye from myFunc...\n";

}

Output from a sample run of the program (user input is in **bold**):

Enter dynamic array size: **3**

0

3

6

How much more dynamic array space do you want? **2**

0

5

10

15

20

Invalid index in show

Invalid index in set

Cannot set - DynArray empty

Cannot show - DynArray empty

hi from myFunc...

0

1

4

9

16

bye from myFunc...

hi from the DynArray destructor...

**What you need to turn in:**

1. **Source code listing:** A printed copy of the source code for each problem. Remember to include the name of each group member in a comment at the top of your source code. Be sure to follow the “Code Style Guidelines” specified in class.
2. **Source code files:** E-mail me your source code as attachments.
3. **Working in Groups:** Every student will work with two other students in our class on this assignment; all members of the group must contribute to the solution. **Turn in only one copy of the solution** – **clearly identify the name of each member on everything that you turn in.**
4. **Late Assignments:** Assignments are due before class on the specified due date (both the paper copies and the e-mail copies). If you wish to turn in the paper copy of an assignment after class, place them under my office door. Assignments turned in late will be assessed a 20% penalty per class day late.