CIS 200 - Lab 0401

1. Problem Statement

Finish the given example of a custom/overridden <vector> class.

2. Requirements

2.1 Assumptions

- Uses the given header file definition and the start of main given.
- Automatically tests and prints to screen
- Console

2.2 Specifications

- Will fill in given functions
- Will test [set] [print] in main()

3. Decomposition Diagram

- Program
    - Input
        - Technically N/A
        - Test Data is set Inline
    - Process
        - Perform Functions
            - print()
            - set()
    - Output
        - Print results to screen

4. Test Strategy

- Valid Data
- Invalid Data

5. Test Plan Version 1

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Created / Not Initialized				
Valid Data	2	Created / Initialized				
Valid Data	3	Set Value 1, Position 0				
Valid Data	4	Set Value 2, Position 1				
Valid Data	5	Set Value 3, Position 2				
Valid Data	6	Set Vector C to D (Deep Copy)				
Valid Data	7	Set Value 0, Position 1				
Valid Data	8	Redefined D, Print				
Valid Data	9	Deep Copy EMPTY				
Invalid Data	1	Set Value 0, Positiion -1				
Invalid Data	2	Set Value 4, Position 3				
Invalid Data	3	Set Size < 0				

6. Initial Algorithm

1. Create Class Vector
    - *a.* Create Default Constructor
        - i. Set size = 0
        - ii. Set entries to new int array/vector[size]
    - b. Create Size Constructor
        - i. If size < 0
            1. ERROR
        - ii. Else
            1. Set size
            2. Set entries to new int array/vector[size]
            3. For (i = 0; i < size; i++)
                a. Fill entries with 0
    - *c.* Create Copy Constructor

        i. Parameter: Constant Vector *other*
        ii. Set size = *other* size
        iii. Set entries to new int array/vector[size]
        iv. For (i = 0; i < size; i++)
            1. Fill entries with *other* entries
    d. Create Default Destructor
    e. Create Function Print
        i. Print First Bracket
        ii. Iterate and Print Each Index
        iii. Print Close Bracket
    f. Create Function Set
        i. If Position is out of bounds
            1. ERROR
        ii. Else
            1. Entry at Pos is Value
2. Create Main
    a. Define Vectors a, b size 3, c size 3
    b. Print a
    c. Print b
    d. Set c (0, -1)
    e. Set c (1, 0)
    f. Set c (2, 1)
    g. Set c (3, 2)
    h. Set c (4, 3)
    i. Print c
    j. Define Vector D as C
    k. Print d
    l. Set d (0, 1)
    m. Print d
    n. Print c

7. Test Plan Version 2

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Created / Not Initialized	a.print()	[ ]		
Valid Data	2	Created / Initialized	b(3), b.print()	[0 0 0]		
Valid Data	3	Set Value 1, Position 0	c.set(1,0)	1 @ 0		

Valid Data	4	Set Value 2, Position 1	c.set(2,1)	2 @ 1		
Valid Data	5	Set Value 3, Position 2	c.set(3,2)	3 @ 2		
Valid Data	6	Set Vector C to D (Deep Copy)	Vector d(c), d.print()	[1 2 3]		
Valid Data	7	Set Value 0, Position 1	d.set(0,1)	0 @ 1		
Valid Data	8	Redefined D, Print	d.print()	[1 0 3]		
Valid Data	9	Deep Copy EMPTY	e(a), e.print()	[ ]		
Invalid Data	1	Set Value 0, Positiion -1	c.set(0,-1)	Error Message		
Invalid Data	2	Set Value 4, Position 3	c.set(4,3)	Error Message		
Invalid Data	3	Set Size < 0	f(-1)	Error Message		

8. Code

[vector.h]

```
#pragma once
#include <iostream>

#ifndef VECTOR_H
#define VECTOR_H

class Vector
{
public:
	//Default Contructor
	Vector();

	//Size Constructor
	Vector(int s);                          //Makes size = s
	                                              //allocate s space
	                                              //     e.g. entries = new
int[size],
	                                              //makes all entries 0

	//Copy Constructor
```

```
	Vector(const Vector& other);//Makes a deep copy

	//Default Destructor
	~Vector();

	void print();			//Prints out the vector
	void set(int val, int pos);	//Store values in vector

private:

	int size;				//Sets the # of elements used
	int *entries;			//Point to array of integers with size
entries
						//	e.g. entries= new
int[size]
};


#endif
```

[vector.cpp]

```
#include "vector.h"
//PUBLIC

//STRUCTOR

//Default Constructor
Vector::Vector()
{
	size = 0;
	entries = new int[size];
}

//Size Constructor
Vector::Vector(int s)
{
	size = s;

	if (size < 0)
	{
		std::cout << "ERROR: SIZE CANNOT BE < 0" << std::endl;
		size = 0;
	}
	else
	{
```

```
		size = s;
		entries = new int[size];

		for (int i = 0; i < size; i++)
		{
			entries[i] = 0;
		}
	}

}

//Copy Constructor
Vector::Vector(const Vector & other)
{
	size = other.size;

	entries = new int[size];

	for (int i = 0; i < size; i++)
	{
		entries[i] = other.entries[i];
	}
}

//DESTRUCTOR
Vector::~Vector()
{
	delete(entries);
}

//END STRUCTOR

void Vector::print()
{
	std::cout << "[";
	for (int i = 0; i < size; i++)
	{
		std::cout << " " << entries[i] << " ";
	}
	std::cout << "]" << std::endl;
	//Prints out the vector
}

void Vector::set(int val, int pos)
{
	if (pos < 0 || pos >= size)
	{
		std::cout << "ERROR: POSITION OR SIZE OUT OF BOUNDS" << std::endl;
```

```
	}
	else
	{
		entries[pos] = val;
	}
}
```

[source.cpp]

```
#include "vector.h"
#include <iostream>

int main()
{
	//REQUIRED CODE

	Vector a, b(3), c(3);

	std::cout << "Valid Test Case 1	: ";
	a.print();		//outputs []
	std::cout << "Valid Test Case 2	: ";
	b.print();		//outputs [ 0 0 0 ]
	std::cout << "Invalid Test Case 1	: ";
	c.set(0, -1);//outputs error message
	c.set(1, 0);
	c.set(2, 1);
	c.set(3, 2);
	std::cout << "Invalid Test Case 2	: ";
	c.set(4, 3); //outputs error message
	std::cout << "Valid Test Cases 3, 4, 5: ";
	c.print();		//outputs [ 1 2 3 ]

	Vector d(c);
	std::cout << "Valid Test Case 6	: ";
	d.print();		//outputs [ 1 2 3 ]

	d.set(0, 1);
	std::cout << "Valid Test Case 7	: ";
	d.print();		//outputs [ 1 0 3 ]
	std::cout << "Valid Test Case 8	: ";
	c.print();		//outputs [ 1 2 3 ] proves deep copy

	//ADDITIONAL TEST CASES
	std::cout << "Valid Test Case 9	: ";
	Vector e(a);
	e.print();
```

```
	std::cout << "Invalid Test Case 3     : ";
	Vector f(-1);

	system("pause");

	return 0;
}
```

---

9. Updated Algorithm

1. Create Class Vector
    a. Create Default Constructor
        i. Set size = 0
        ii. Set entries to new int array/vector[size]
    b. Create Size Constructor
        i. If size < 0
            1. ERROR
        ii. Else
            1. Set size
            2. Set entries to new int array/vector[size]
            3. For (i = 0; i < size; i++)
                a. Fill entries with 0
    c. Create Copy Constructor
        i. Parameter: Constant Vector *other*
        ii. Set size = *other* size
        iii. Set entries to new int array/vector[size]
        iv. For (i = 0; i < size; i++)
            1. Fill entries with *other* entries
    d. Create Default Destructor
    e. Create Function Print
        i. Print First Bracket
        ii. Iterate and Print Each Index
        iii. Print Close Bracket
    f. Create Function Set
        i. If Position is out of bounds
            1. ERROR
        ii. Else
            1. Entry at Pos is Value
2. Create Main
    a. Define Vectors a, b size 3, c size 3

b. Print a
c. Print b
d. Set c (0, -1)
e. Set c (1, 0)
f. Set c (2, 1)
g. Set c (3, 2)
h. Set c (4, 3)
i. Print c
j. Define Vector D as C
k. Print d
l. Set d (0, 1)
m. Print d
n. Print c
o. Define Vector E as A
p. Print e
q. Attempt to Define f with size -1

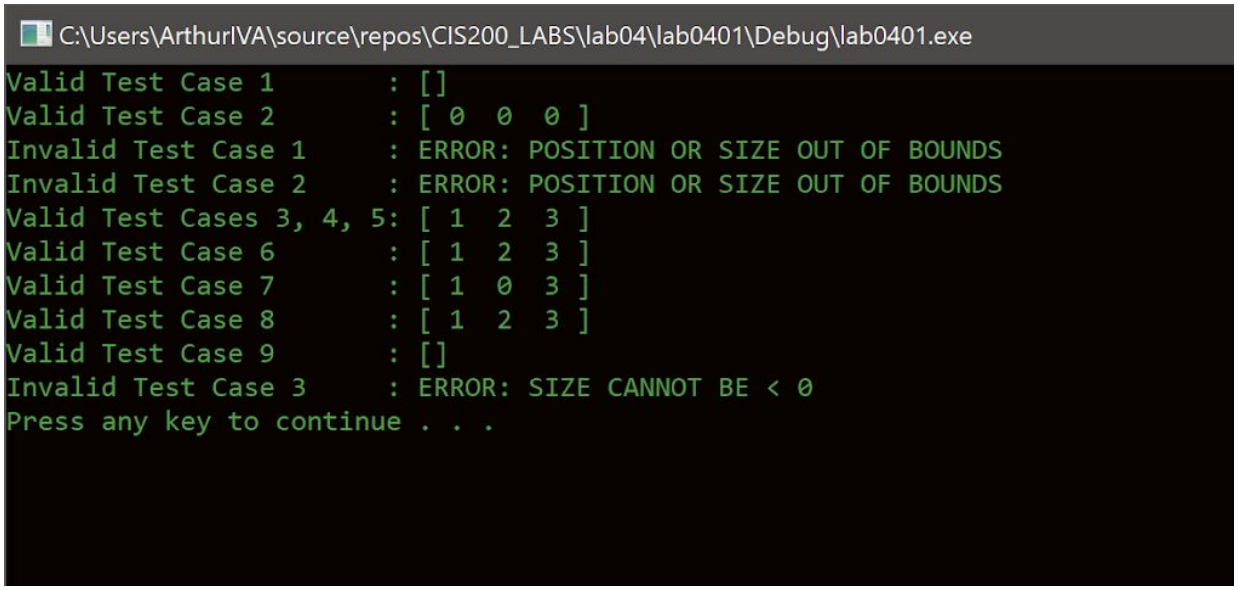
3.

10. Test Plan Version 3

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Created / Not Initialized	a.print()	[ ]	[ ]	[ ]
Valid Data	2	Created / Initialized	b(3), b.print()	[0 0 0]	[0 0 0]	[0 0 0]
Valid Data	3	Set Value 1, Position 0	c.set(1,0)	1 @ 0	1 @ 0	1 @ 0
Valid Data	4	Set Value 2, Position 1	c.set(2,1)	2 @ 1	2 @ 1	2 @ 1
Valid Data	5	Set Value 3, Position 2	c.set(3,2)	3 @ 2	3 @ 2	3 @ 2
Valid Data	6	Set Vector C to D (Deep Copy)	Vector d(c), d.print()	[1 2 3]	[1 2 3]	[1 2 3]
Valid Data	7	Set Value 0, Position 1	d.set(0,1)	0 @ 1	0 @ 1	0 @ 1
Valid Data	8	Redefined D, Print	d.print()	[1 0 3]	[1 0 3]	[1 0 3]
Valid Data	9	Deep Copy EMPTY	e(a), a.print()	[ ]	[ ]	[ ]
Invalid Data	1	Set Value 0, Positiion -1	c.set(0,-1)	Error Message	Error Message	Error Message

Invalid Data	2	Set Value 4, Position 3	c.set(4,3)	Error Message	Error Message	Error Message
Invalid Data	3	Set Size < 0	f(-1)	Error Message	Error Message	Error Message

11. Screenshots



12. Error Log

Error Type (Logic/Runtime)	Cause of Error	Solution to Error
Logic	Not setting size at the beginning of size constructor	Add that!

13. Status

Program is alive and well. Except it's allergic to gluten.