CS 202: Computer Science II
Assignment 5: Box of Chocolates

"My mom always said life was like a box of chocolates.
You never know what you're gonna get." - Forrest Gump

Computer Science programming assignments are like a box of chocolates. You never know what you are gonna get and today you're getting a Forrest Gump themed assignment! Very meta ya?

In this assignment you will develop your skills with pointers further by exploring the use of inheritance and virtual functions as you create a 2D array of pointers to recreate Forrest Gump's famous box of chocolates. When finished the program will replicate the concept of grabbing a full box of chocolates and selecting randomly and removing, or eating, one chocolate at a time. This process will be continued until you randomly select a location in the box that no longer has a chocolate. At which point you will stop eating chocolates. You can think of the box of chocolates as a 2D array where each position can hold a chocolate. There is an assortment of different types of chocolates that each give you a unique response that is printed to the screen. For example, whenever you take a mint flavored chocolate. Your character will wonder whether he is still eating chocolate or if he is brushing his teeth. This is as far as the program will go but really each of these chocolates could do a lot of other unique actions, but we want to keep it simple to focus on practicing the new topics which are virtuals and multi-dimensional dynamic arrays.

There are 8 classes in the program. These are distributed as follows: First there is a *superclass* called **chocolate**. Next there are six other *subclasses* that are derived from that superclass. These represent different types, or flavors, of chocolates (class names: **dark, milk, hazelnut, raspberry, white,** and **mint**). Finally, there is one class that represents the box of chocolates. called, wait for it, **boxOfChocolates**. If we were not using Codegrade I would let you all choose creative flavor names, but that would make autograding it complicated so please stick to these classnames/flavors.

In addition to the classes, there is the main function whose job is to read in a command line argument representing the seed of the program. This seed value is an integer that will be used to generate the RNG of the box, so each box of chocolate is unique in combination of flavors!

Let's look closer at the required functions for each of the classes.

Class Name: **chocolate**
Functions: One pure virtual function called **whatamI()** that can return a string.

Nesxt is the 6 classes that inherit from chocolate. Note that each one will define the virtual function inherited from chocolate by returning as a string the name of that class. For example, class dark will return "dark" in whatamI. This will be useful when you need to figure out what the pointer is really pointing to so you can downcast to the correct class.
In addition each class will have one unique function that just couts some predefine statements that are as follows:

Class Name: **dark**
```
Function: void feelBitter(){
        cout << "Dark Chocolate: oof that was bitter!" << endl;
    }
```

Class Name: **milk**
```
Function: void feelSoft(){
        cout << "Milk Chocolate: Too soft, no chocolate taste!" << endl;
    }
```

Class Name: **hazelnut**
```
Function: void getAllergy(){
        cout << "Hazelnut Chocolate: I don't feel so good." << endl;
    }
```

Class Name: **raspberry**
```
Function: void telljoke(){
        cout << "Raspberry Chocolate: I don't always Raspberry,"
            << " but when I do, I Raspberry Pi" << endl;
    }
```

Class Name: **white**
Function: void feelScammed(){
```
      cout << "White Chocolate: Is this even real chocolate?" << endl;
  }
```

Class Name: **mint**
Function: void feelInquisitive(){
```
      cout << "Mint Chocolate: Am I still eating chocolate"
          << " or am I brushing my teeth?" << endl;
  }
```

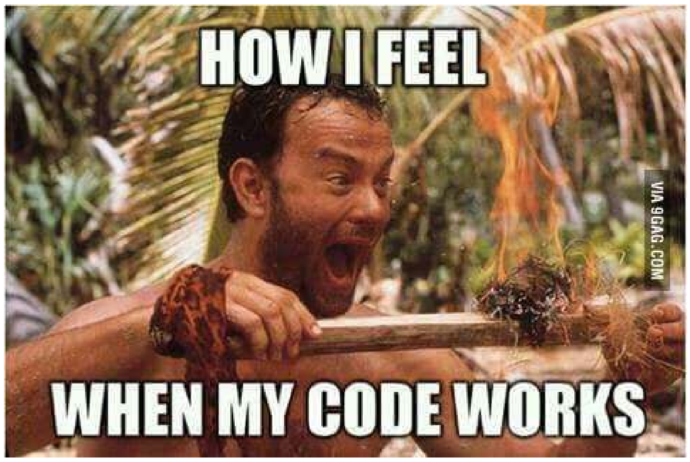
Finally let's look at class **boxOfChocolates**. This class requires a constructor, a destructor, and two public functions void fillBox() and bool takeChocolate(). It also contains three private variables: Two integers boxLength and boxWidth and a pointer (***) of chocolate type named box. The first two variables obviously keep track of the dimension of the box and the pointer variable is used to dynamically allocate a 2D array of pointers via the constructor. As you might have guessed it, you de-allocate the box in the destructor. Your constructor should have default parameters of 6 for length and 5 for width. But allow the user to pass those in that order if they wish to build different sized boxes. After allocating the box in the constructor and setting the two integers with the length and width passed in (or using the defgault), have the constructor call the fillBox function.

The fillBox function's job is to go through the entire array of pointers (so the box) and fill it randomly with chocolates of different flavors. So your randomness matches codegrade. Here is what I did. I just called rand() in a switch statement using % 6 for each of the flavors. The order of the cases from 0-5 was dark, milk, hazelnut, raspberry, white, and mint. As long as you do this then your RNG will match the codegrade's RNG for a given seed value. Make sure the fillBox function fills the entire fox.

Finally the takeChocolate function calls rand() twice to get first a length and then a width (make sure you match this order so RNG matches codegrade) and then that is the spot that will be removed (chocolate taken) from the box. To do so you will need to use the whatamI function to downcast the chocolate pointer to the right subclass, and then go ahead and call the unique function for each class so it couts/prints its statement accordingly. Theoretically since all we are doing in each function is printing statements, you could have used polymorphism to do this all in a single function without casting, but then you wouldn't practice downcasting. So just imagine the functions do more than cout stuff, so then you actually did need to use this system. If you don't do it like that then you won't get full points. After you run that function code, go ahead and delete /de-allocate that spot in the array. Don't forget to set it to null to avoid dangling pointers! This function will return true if a chocolate is removed, otherwise it will return false if the location chosen by RNG already had an empty/null spot meaning that a chocolate from that spot was already removed.

Your main function should then consist of printing out the Forrest gump quote (see sample output, match it), running srand with the integer in the command line argument (I recommend using atoi to do the string->int conversion), allocating an object of boxOfChocolates, and then running an infinite loop that just calls box.takeChocolate() repeatedly until the given function returns false.

The final product should show anywhere from removing all chocolates and printing a statement for each, to potentially removing only one chocolate and ending up picking the same spot again resulting in that single cout statement. Obviously it will trend to be more in the middle, but do share on Discord if you find seeds that have both extreme cases as we can all test them out!



So that's the assignment. And that's all I have to say about that.

Notes:
-Make sure your code passes all codegrade tests if you want to receive full points.
-Test your program by running it a few times with different seeds to make sure hit all potential cases.
-Make sure your output is EXACTLY formatted like the given sample outputs or codegrade will eat your grade.
-Comment your source code appropriately according to the stipulations on the rubric.
-**Your program must have 0 errors from 0 contexts, and 0 bytes in 0 blocks definitely, indirectly, or possibly lost in valgrind.** That means **NO MEMORY LEAKS** allowed.

Here is some sample output. Note I added an extra check for when no argument is passed to avoid a segfault. I recommend you do the same, but we will not check for that with codegrade as it's not the focus of the assignment..

Sample Output to Terminal (note the different seeds I feed, they should match yours):

```
AST05$ g++ main.cpp
AST05$ ./a.out
Enter Seed as argument: ./a.out 42
AST05$ ./a.out 42
"My mom always said life was like a box of chocolates. You never know what you're
gonna get." - Forrest Gump
White Chocolate: Is this even real chocolate?
Dark Chocolate: oof that was bitter!
Dark Chocolate: oof that was bitter!
Dark Chocolate: oof that was bitter!
Mint Chocolate: Am I still eating chocolate or am I brushing my teeth?
Dark Chocolate: oof that was bitter!
Raspberry Chocolate: I don't always Raspberry, but when I do, I Raspberry Pi
White Chocolate: Is this even real chocolate?
Mint Chocolate: Am I still eating chocolate or am I brushing my teeth?
Mint Chocolate: Am I still eating chocolate or am I brushing my teeth?
AST05$ ./a.out 7
```

```
"My mom always said life was like a box of chocolates. You never know what you're
gonna get." - Forrest Gump
Dark Chocolate: oof that was bitter!
Mint Chocolate: Am I still eating chocolate or am I brushing my teeth?
Raspberry Chocolate: I don't always Raspberry, but when I do, I Raspberry Pi
AST05$ ./a.out 9001
"My mom always said life was like a box of chocolates. You never know what you're
gonna get." - Forrest Gump
Mint Chocolate: Am I still eating chocolate or am I brushing my teeth?
Mint Chocolate: Am I still eating chocolate or am I brushing my teeth?
Dark Chocolate: oof that was bitter!
Hazelnut Chocolate: I don't feel so good.
Hazelnut Chocolate: I don't feel so good.
Raspberry Chocolate: I don't always Raspberry, but when I do, I Raspberry Pi
AST05$
```