Name: _____ ID: _____

Object-Oriented Programming Lab #5 **Feb 17th, 2023**

User Defined Types

1. Write a class for representing a book information along with basic operations and test programs.

1.1) Write a `Book` class, such as you can imagine as part of software for a library.
Required data: ISBN, title, author, and copyright year
You are required to:

- Provide appropriate **constructors** for class `Book`
- Provide appropriate **accessors** for each data of a `Book` object
- Write a test program for testing all use cases of a `Book` object and its operations

The ISBN must be in the form of **n-n-n-x** where **n** is an integer and **x** is a digit or a letter.

1.2) From **1.1)** add the following operations:

- **Check in** and **check out** a book
- Checking whether or not the book is checked out (**available**)
- **Compare** the ISBN numbers for two books and check whether they are the same (**is equal/same**)
- **Print** information about a book to the output stream

Write a test program for testing all of the above operations.

1.3) Modify the code from **1.2)** by creating an enumerated type for the `Book` class called `Genre`. The `Genre` type will define values for **fiction**, **nonfiction**, **periodical**, **biography**, and **children**. Adding the `Genre` information to a `Book` object by making appropriate changes to the class `Book`. Finally, write a test program for testing added features.
Advice: Use scoped enumeration (`enum class`) to define a set of named integral constants

1.4) Modify the code from **1.3)** by adding the following operations:

- **Read** a book data from the input stream
  The data format will be `"<ISBN>\n<title>\n<author>\n<copyright>\n<genre>\n"`, each consumes the whole line of text
- **Find** a book that match the ISBN from a collection of books

Write a test program for testing all of the above operations.

1.1)		1.2)		1.3)		1.4)	

2. Write a class for representing an ASCII picture along with basic operations and test programs.

2.1) Write a `Picture` class which stores a collection of rows of a text string for its content. The longest row determines the width and the number of rows represents the height.
You are required to:

- Provide appropriate **constructors** for class `Picture`
- Provide appropriate **member functions** for getting the width and the height of a `Picture` object
- Write a test program for testing all use cases of a `Picture` object and its operations

2.2) From **2.1)** add the following operations:

- `hflip` for **flipping** the contents of a picture **horizontally**, if the row is shorter than the width, fill blank characters on the right to fit the width before flipping the contents
- `vflip` for **flipping** the contents of a picture **vertically**
- **Print** the contents of a picture to the output stream

Write a test program for testing all of the above operations.

2.3) Modify the code from **2.2)** by adding the following operations:

- `hcat` for creating a new picture by **concatenating** two pictures **horizontally**
- `vcat` for creating a new picture by **concatenating** two pictures **vertically**

Add additional support operations as needed. Write a test program for testing all of the above operations.

2.4) Modify the code from **2.3)** by adding a function `resize` to adjust the width and height of a picture. The function will expand the picture size when the new width/height is larger and crop the picture when the new width/height is smaller. Add additional support operations as needed. Finally, write a test program for testing the function.

2.1)	**2.2)**	**2.3)**	**2.4)**

3. Given the following SVG image file as a template:

```
<svg width="100" height="100" xmlns="http://www.w3.org/2000/svg">
  <style type="text/css">
    rect { stroke: #00FFFF; }
    rect.c1 { fill: #000000; }
    rect.c2 { fill: #808080; }
  </style>

  <rect class="c1" x="10" y="10" width="20" height="20" />
  <rect class="c1" x="30" y="10" width="20" height="20" />
  <rect class="c2" x="50" y="10" width="20" height="20" />
  <rect class="c2" x="70" y="10" width="20" height="20" />
</svg>
```

and the following example XPM file representing a checker pattern image:

```
! XPM2
4 4 2 1
# c #000000
- c #FFFFFF
##--
##--
--##
--##
```

- `"! XPM2"` is the header for identifying XPM2 format
- `"4 4 2 1"` in the file means that the image has both the **width** and the **height** of 4 pixels, 2 **colors** and 1 character per pixel.
- The next two lines (`"<c> c <color>"`) map a character `<c>` to the color `<color>` which means `'#'` is black and `'-'` is white
- The rest is the image contents

3.1) Extend or rewrite the `Picture` class from **exercise 2** to write a program that generates an SVG image that visualize the contents of the `Picture` content.
You are required to:

- Write a function `gen_svg(pic, os)` that can be used to generate an SVG image from the `Picture` object `"pic"` and write the output to the output stream `"os"`
  - Each rectangle will represent each character/pixel in the `Picture`
  - Use random colors for pixel values
  - Use the same color for the same pixel in the `Picture` contents
- Write a test program for testing `gen_svg` function and check if the output SVG file is correct
- Write a test program that uses all operations from **exercise 2** to compose `Picture` objects and generate final SVG output

Add additional support operations as needed.

3.2) Modify the code from **3.1)** to provide more features.

You are required to:

- Add color map data to the `Picture` class
- Add operations for setting the color map data of a `Picture` object
- Add operations for reading an XPM file from the input stream and create a `Picture` object
- Add operations for writing a `Picture` object to an XPM file through the output stream
- Modify a function `gen_svg(pic, os)` to use the color map data from a `Picture` object to map color for each rectangle representing a pixel in the `Picture`

Add additional support operations as needed. Write a test program for testing all of the above features.

3.1)

3.2)