Assignment 1 - Replacement: Responsive Layouts

Goals for this CW:

- Demonstrate a working understanding of writing custom layouts in C++ and Qt
- Create a prototype layout for an app
- Design a beautiful responsive layout

Getting started:

- Download and extract the zip file
- Open the *.pro* file with Qt Creator
    - For lab linux:
        - module load legacy-eng
        - module add qt/5.15.2
        - module add gcc
        - qtcreator responsive.pro
- Run the project. It creates a main window with several coloured labels which can be manually resized. Try changing the size of the window; observe the responsive design - the number, location and size of the widgets change - but the design is terrible.
- Run the project in automatic test mode (**warning your screen will flash rapidly**):

    1. Add a command line argument: *test* (without quotes).
    2. Add another command line argument: a valid path where a folder called “report” will be created. Use quotation marks around the path.
    3. Run the program again.
    4. Observe that a number of different window sizes are displayed, screenshots are taken, and written to the *report* directory in the root of the project. The program then exits. View the *report/index.html* in a browser and observe the terrible responsive design.
    5. Set the project back to manual mode for your development by removing the program argument *test*.

- The project contains three important C++ classes:

    1. `ResponsiveWindow`: This subclasses `QWidget` and creates the widgets to be displayed in the `ResponsiveWindow::createWidgets` function. It sets a `ResponsiveLayout` and adds various `ResponsiveLabel`s.
    2. `ResponsiveLayout`: This subclasses `QLayout` and arranges the `ResponsiveLabels` in the window using the `ResponsiveLayout::setGeometry` function.
    3. `ResponsiveLabel`: This subclasses `QLabel` and creates a label widget with a name and a coloured background.

- Read the classes to understand how they work together to create the responsive layout you have observed.

Your task:

You will create a *prototype layout* for a page of a music-focused social network. This prototype will demonstrate which widgets are shown on what sized devices and how their layout adapts in a responsive manner. You will work on the “Personal profile” webpage.

The following widgets and associated colours have been predefined in the *responsive_label.h* file. This is what would be their expected function in the context of this page.

1. Home: Takes you to the “home” of the social network where you see everyone’s posts.
2. Navigation Area: container area where several top level links are usually placed. You can overlap items on top of this.
3. My Profile: Takes you to “your profile”, the current page.
4. Favorite Track: Name of your favorite track and artist.
5. Events: Link to a page that shows music events in your area.
6. Profile picture: Your photo.
7. Album art: photo of album cover of a given song.
8. Song name: -name of given song.
9. next page of search results button (`kSForward`)
10. previous page of search results button (`kSBackward`)

The main things this page should always display are:
- A chronological list of the songs you have most recently listened to.
- Your photo.

Specific tasks & Marking criteria:

Implement a responsive layout, where widgets are appropriately placed according to different window sizes.

(6 Marks)

Widgets should not overlap and do not extend beyond the edge of the window.

(3 Marks)

Adapt the size of the labels according to their function, trying to allow for readability in all possible window sizes.

(3 Marks)

Ensure the Layout has beautiful, non-zero, and appropriate spacing between the widgets without any large gaps.

(3 Marks)

Add up to two additional label types of your own and include them in your layout.

(2 Marks)

Implement two extra alternative layouts (to a total of 3). One Horizontal and one vertical layout which should be picked according to the aspect ratio of the screen, and a “compact” layout”, when the window is too small to hold most of your labels. Everything should still be responsive.

(6 Marks)

Research and use `QScrollArea` to allow the page to scroll and show more recent songs (both album art and song name).

(6 Marks)

Create a layout for your scroll area which positions 17 search results in a responsively sized grid layout. Each whole song item (art and song name) should take up a square of space. The horizontal and vertical spacing between the results, and around the grid, should be consistent.

(6 Marks)

Notes:

- You should carefully plan which labels are shown at different screen sizes and orientations by researching which elements are prioritised at which sizes on real-world shopping apps and websites.
- You may wish to sketch the widgets on paper before you write code.
- Only use `ResponsiveLabel` widgets with a simple text description (and a single `QScrollArea` if you attempt the extension task).
- The widgets don't need to do anything (i.e., signals and slots are not required; nothing is expected to happen when you click on a label).
- It is not necessary for all the label text to always be visible.
- None of Qt's built-in layouts may be used (`QVBoxLayout`…).
- Do not edit the files with the warning "`DO NOT EDIT THIS FILE`" at the top. They will be used during the marking process.
- The window should show a *usable, beautiful,* and *useful* layout whether sized at its minimum (320x320 pixels) or its maximum (1280 x 720 pixels), as well as all sizes in between. Landscape and portrait orientations should be considered.
- The prototype is for a full-screen app with a touch-based interface (for a variety of different sized mobile and tablets devices). The widgets should be positioned and sized accordingly.
- We will take the physical dimension of the minimum size window (320x320 pixels) to be 4.5x4.5 cm (about the size of a small mobile phone screen). Larger physical dimensions are taken to be scaled proportionally.
- Avoid large gaps in any layouts. Space after a final search result, e.g., to finish the row, is acceptable.
- It is suggested to add the maximum number of ResponsiveLabels required for all possible layouts in `ResponsiveWindow::createWidgets` and hide them as required in `ResponsiveLayout::setGeometry`. Widgets can be hidden by setting their size to zero.

To submit:

- Zip your project and submit to gradescope by the deadline at the bottom of this document.
- Your folder should contain **only** your source code and the .pro file.

- You may submit multiple times. Only the most recent will be graded. Please submit early and often

DISCLAIMER:

Other than Qt and the codebase you downloaded with this assignment, ALL code must be written by you personally. While discussing solutions with colleagues is allowed, sharing code is forbidden.

You **must** implement all of the functions above using the framework provided, as working within an existing codebase is a valuable skill to master, and will be graded in this coursework.

The code **MUST** run on the University's Linux system. You may implement on your own machine, but it will be tested on ours.

PENALTIES:

Every function you implement should contain comments using your own words explaining what you did. This will be used to differentiate your work from your colleagues, and to evaluate your understanding of what you did. Poorly commented submissions may be penalised by up to 25% of the marks available.

Poorly structured code may be penalised by up to 25% of the marks available.

Code that does not compile properly will be assigned a mark of 0