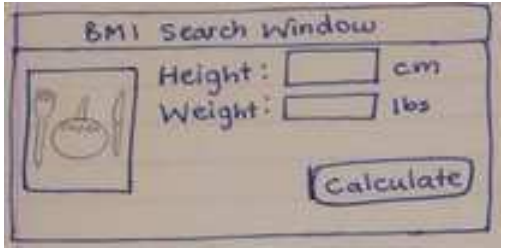
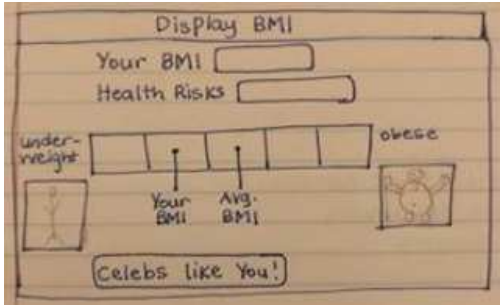
Specification

This GUI uses an API to calculate a persons Body Mass Index after they enter their height and weight. It also tells them whether they are underweight or overweight and about potential risks they could face. Depending on the output, the GUI will open a window that shows celebrities who are in the same BMI range as the user.

Analysis

We will use the mashape marketplace to acquire an API that calculates the user's BMI. (https://market.mashape.com/nviror/bmicalculator) Users input their measurements in the appropriate search box (weight in pounds and heightin inches). The API does not process inches so we will convert it to meters before passing it to the API. The information is returned in json format. We are going to use this API to parse it: https://github.com/nlohmann/json. Once we find the url from that info, we can use the CURL API to retrieve the users BMI. https://curl.haxx.de. The BMI will be displayed on a new window that will also show the average BMI of previous users and a button labelled **Celebs with Similar BMIs**. The button will open a new window with celebrities who are in the same BMI range as the user. Below is an early protoype of out GUI.





Design

- main
  - Show the BMI User's input Window
- makeInputWindow
  - Creates Input Window for User's search
- findCorrectWindow
  - Refer to output window from input window using callback function
- cbHeight
  - Converts inches to meters
- searchAPI
  - Retreive BMI information from API
- getAPIInfo
  - Parse information to find relevant values
- makeOutWindow
  - Creates an output that shows BMI, User Status, and Risks
- cbCeleb
  - Calls CelebWindow once the user presses a button
- CelebWindow
  - Creates a window that shows pictures of Celeberties that have similar BMI as the user
- displayGif
  - Calls functions needed to run gif
- loadimages
  - Labels each frame and stores them in an array
- animateGif
  - Animates gif by loading images in the correct sequence
- lab.h
  - List of declarations and FLTK functions used

Implementation: main.cpp

```
#include "lab.h"

int main()
{
    makeInputWindow()->show();
    Fl::run();
}
```

- Run and show makeInputWindow
- If any windows are open, calls Fl::wait repeatedly
  - Runs the program everytime there is a change
  - i.e. If the user puts in a new weight or height

Implementation: makeInputWindow.cpp

```
#include "lab.h"
#include <math.h>
void makeOutWindow(Fl_Button*,void*);
void findCorrectWindow(Fl_Button*,void*);
Fl_Cairo_Window * cw;
Fl_Input * wi; Fl_Input * hi;
Fl_Button * t; Fl_Box * ii;
Fl_Cairo_Window* makeInputWindow()
{
    const int w = 600;
    const int h = 600;
    cw = new Fl_Cairo_Window(w,h);

    cw->label("BMI Search");
    cw->color(FL_CYAN);
    wi = new Fl_Input(.33*w,365,.45*w,.1*h,
                      "Weight (lb)");

    hi = new Fl_Input(.33*w,435,.45*w,.1*h,
                       "Height (in)");

    t = new Fl_Button(.33*w,505,.45*w,.1*h,
                      "Calculate");
    t->callback((Fl_Callback*)findCorrectWindow);

    ii = new Fl_Box(50,50,500,281);
    displayGif();
    return cw;
}
```

- Make new input window
- Add weight and height input
- Button calls **findCorrectWindow**
- Make a box to display the gif in
- Call **displayGif**

Implementation: findCorrectWindow.cpp

```
#include "lab.h"M̂
BMIInfo bmi;M̂
void findCorrectWindow(Fl_Button*,void*)M̂
{   M̂
    cbHeight();M̂
    bmi = getAPIInfo(); M̂
    if(ow) ow->hide();M̂
    makeOutWindow()->show();M̂
}M̂
```

- Run **cbHeight**
- Run **getAPIInfo**
- If an output window already exists, hide it
- Open a new output Window

Implementation: cbCelebDoc.cpp

```
#include "lab.h"
std::string hiii;
void cbHeight()
{
    double hii;
    std::istringstream iss(hi->value());
    iss >> hii;
    hii= hii * 0.0254;
    hii = (round(hii*100))/100;
    std::ostringstream oss;
    oss << hii;
    hiii = oss.str();


}
```

- Take hi(height input) string and convert to double
- Convert input from inches to meters by multiplying by 0.0254
  - API does not take inches
- Round to two decimal points
  - **round** only rounds off to whole numbers
  - Mutiply input by 100 and then round to a whole number
  - Divide by 100
- Convert double back into string

Implementation: searchAPI.cpp

```
#include "lab.h" M̂
#include <curl/curl.h>M̂
const std::string url = "https://bmi.p.mashape.com";M̂
const std::string key = "X-Mashape-Key: 0PXIy4DqFamshCasU4dEwk47Gso8p1W4mFYjsnTi4u
const std::string type = "Content-Type: application/json";M̂
const std::string js = "Accept: application/json";M̂
std::string data = "";M̂
size_t handleData(void* c, size_t s, size_t n, void* j)M̂
{M̂
    *static_cast<std::string*>(j) += static_cast<char*>(c);M̂
    return s * n;M̂
}M̂
std::string searchAPI(std::string wi,std::string hi)M̂
{M̂
    std::string s = "";M̂
    struct curl_slist* slist1 = NULL;M̂
    slist1 = curl_slist_append(slist1,key.c_str());M̂
    slist1 = curl_slist_append(slist1,type.c_str());M̂
    slist1 = curl_slist_append(slist1,js.c_str());M̂
    std::string q = url;M̂
    data = "{\"weight\":{\"value\":"+ wi +",\"unit\":\"lb\"},\"height\":{\"value\":"+ hiii +",\"unit\":\"m\"},\"sex\":\"m\",\"age\":\"24\",\"waist\":\"34.00\",\"hip\":\
    CURL* hnd = curl_easy_init();M̂
    curl_easy_setopt(hnd,CURLOPT_URL,q.c_str());M̂
    curl_easy_setopt(hnd,CURLOPT_HTTPHEADER,slist1);M̂
    curl_easy_setopt(hnd,CURLOPT_POSTFIELDS,data.c_str());M̂
    curl_easy_setopt(hnd,CURLOPT_WRITEFUNCTION,handleData);M̂
    curl_easy_setopt(hnd,CURLOPT_WRITEDATA,&s);M̂
    curl_easy_perform(hnd); curl_easy_cleanup(hnd);M̂
    return s; M̂
}M̂
```

- Make new output window
- Output BMI value
- Output BMI status
- Output Health Risks
  - **Fl Text Buffer** makes a scrollbar if the text is too long

Implementation: getAPIInfo.cpp

```
#include "lab.h"
#include "json.hpp"
using nlohmann::json;
BMIInfo getAPIInfo()
{
    std::ostringstream oss; oss << wi->value();
    std::ostringstream oss2; oss2 << hi->value();
    std::string s = searchAPI(oss.str(),oss2.str());
    while(s.back()!='}') s.pop_back();
    auto j = json::parse(s);
    BMIInfo bi;
    std::string value = j["bmi"]["value"];
    std::cout << value << std::endl;
    bi.status = j["bmi"]["status"];
    bi.risk = j["bmi"]["risk"];
    std::istringstream iss(value);
    iss >> bi.value;
    return bi;
}
```

- Get string **s** with BMI information from searchAPI.cpp
- Use **popback** to truncate any extra characters
- Use json library to parse string
  - Find BMI value, user status, health risk
- Convert BMI value from string to double

Implementation: makeOutWindow.cpp

```
#include "lab.h"
Fl_Cairo_Window * ow; Fl_Button * c; Fl_Box * gb; Fl_Output * st;
Fl_Value_Output * bm; Fl_Text_Display * hr;
Fl_Cairo_Window* makeOutWindow()
{
    ow = new Fl_Cairo_Window(650,650);
        ow->label("Your BMI Information"); ow->color(FL_CYAN);
    bm = new Fl_Value_Output(.2*w,.02*h,.25*w,.05*h, "Your BMI");
        bm->color(FL_WHITE); bm->value(bmi.value);
    st = new Fl_Output(.6*w,.02*h,.5*w,.05*h, "Status");
        st->value(bmi.status.c_str());
    hr = new Fl_Text_Display(.2*w,.1*h,.92*w,.07*h, "Health Risks");
        hr->buffer(new Fl_Text_Buffer());
        hr->buffer()->text(bmi.risk.c_str());
    gb = new Fl_Box(69,150,512,384);
    std::string z = bmi.status;
    if (z == "Severely underweight"){
            gb -> image(new Fl_JPEG_Image("u.jpg"));
        } else if (z == "Underweight"){
            gb -> image(new Fl_JPEG_Image("u.jpg"));
        } else if (z == "Normal (healthy weight)"){
            gb -> image(new Fl_JPEG_Image("h.jpg"));
        } else if (z == "Overweight"){
            gb -> image(new Fl_JPEG_Image("ov.jpg"));
        } else if (z == "Obese Class I (Moderately obese)"){
            gb -> image(new Fl_JPEG_Image("o1.jpg"));
        } else if (z == "Obese Class II (Severely obese)"){
            gb -> image(new Fl_JPEG_Image("o1.jpg"));
        } else if (z == "Obese Class III (Very severely obese)"){
            gb -> image(new Fl_JPEG_Image("o2.jpg"));}
    c = new Fl_Button(.46*w,.95*h,.4*w,.05*h,
                        "Celeberities Like You");
    c->callback((Fl_Callback*)cbCeleb); c->color(FL_WHITE);
    return ow;
}
```

- Make new output window
- Displays BMI value, User Status, and Health Risks
  - **Fl Text Buffer** makes a scrollbar if the text is too long
- Use **if else** statements to choose an image that matches the users' status
- Has a button linked to **cbCeleb**

Implementation: CelebWindow.cpp

```
#include "lab.h"
Fl_Box * cb; Fl_Cairo_Window * b;
Fl_Cairo_Window* CelebWindow()
{
    b = new Fl_Cairo_Window(600,600);
    b->label("Celebrities in the same BMI Range");
    b->color(FL_CYAN);
    {
    cb = new Fl_Box(0,0,600,600);
    std::string z = bmi.status;
    std::cout << "celeb window:" << z << std::endl;
        if (z == "Underweight"){
            cb->image(new Fl_JPEG_Image("underweight.jpg"));
        } else if (z == "Normal (healthy weight)"){
            cb -> image(new Fl_JPEG_Image("healthy.jpg"));
        } else if (z == "Overweight"){
            cb -> image(new Fl_JPEG_Image("overweight.jpg"));
        } else if (z == "Obese Class I (Moderately obese)"){
            cb -> image(new Fl_JPEG_Image("obese1.jpg"));
        } else if (z == "Obese Class II (Severely obese)"){
            cb -> image(new Fl_JPEG_Image("obese1.jpg"));
        } else if (z == "Obese Class III (Very severely obese)"){
            cb -> image(new Fl_JPEG_Image("obese2.jpg"));
        }
    }
    return b;
}
```

- Create new window for celebrity information
- Make a new box to put appropriate image
- Use **if else** statements to find correct image
  - Compares BMI status to callback an image

Implementation: cbCelebDoc.cpp

```
#include "lab.h"
void cbCeleb(Fl_Button*,void*)
{
    if(b) b->hide();
    CelebWindow()->show();
}
```

- If a CelebWindow already exists, hide it
- Show CelebWindow

Implementation: displayGif.cpp

```
#include "lab.h"
void displayGif()
{
    loadimages();
    Fl::add_timeout(1.0,animateGif);



}
```

- Call **loadimages**
- Call **animateGif**

Implementation: loadimages.cpp

```
#include "lab.h"
const int FRAMES = 31;
Fl_GIF_Image* gifs[FRAMES];
void loadimages()
{
    for(int i = 0; i < FRAMES; i++)
        {
            std::ostringstream oss;
            oss << std::setfill('0') << std::setw(3) << i;
            std::string s = "gif/g" + oss.str() + ".gif";
            gifs[i] = new Fl_GIF_Image(s.c_str());
        }
}
```

- Create an array to hold each frame
- For loop to label and load images
  - Labels each image with three digits in chronological order
    - i.e. first frame would be gif/g001.gif, the second would be gif/g002.gif,etc
  - Convert to string with oss.str()
    - **oss.str()** extracts string from stringstream
    - **c.tr()** extracts c string from c++ string
  - Loads appropriate image

Implementation: animateGif.cpp

```
#include "lab.h"
void animateGif(void*)
{
    static int i = 0;
    ii->image(gifs[i]);
    i = (i + 1) % 31;
    cw->redraw();
    Fl::repeat_timeout(1.0/30,animateGif);
}
```

- Initializes loop
- Uses modulus function to loop the frames
  - Add 1 because the first frame is labeled 000 not 001
  - Shows frames until i is 0
  - Loops back to first frame of gif
- Repeats the loop 30 frames per second

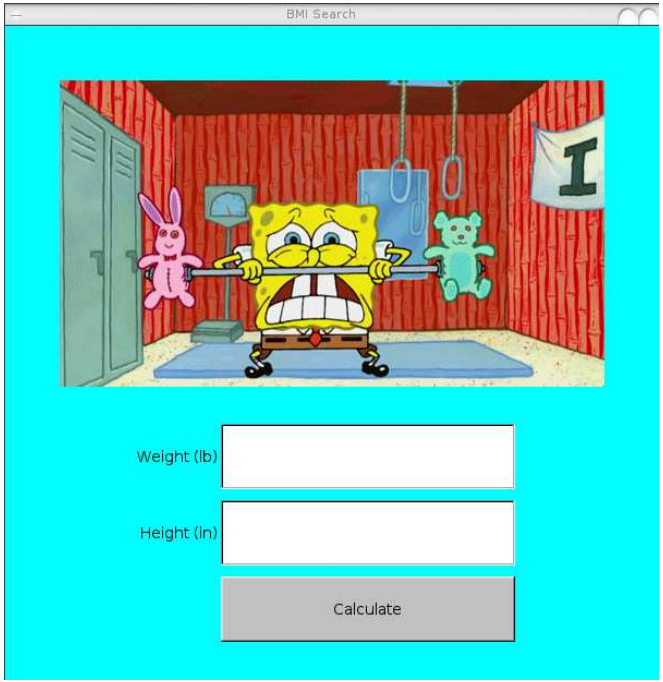
Implementation: lab.h

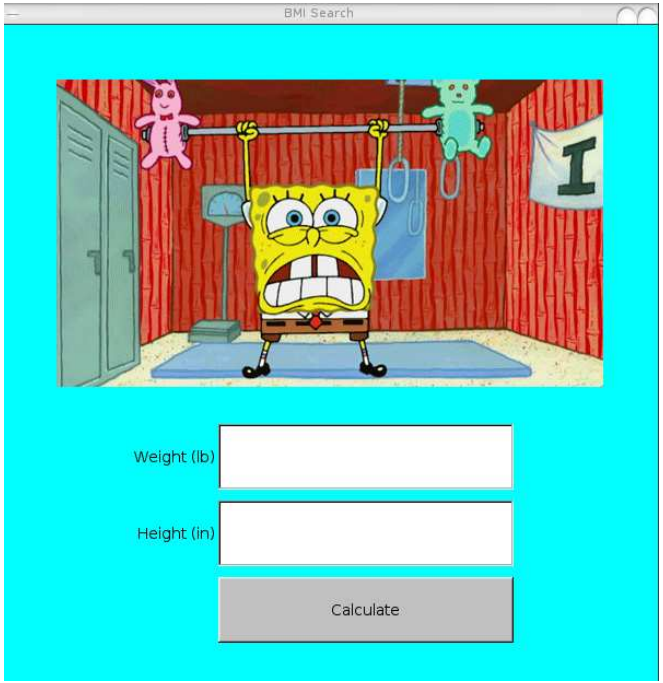
- List of all the FLTK functions used
- List of declarations and variables
- BMI struct

```
#include <iostream>
#include <sstream>
#include <cmath>
#include <string>
#include <cstdlib>
#include <config.h>
#include <iomanip>
#include <cairo-xlib.h>
#include <FL/Fl_Cairo_Window.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_GIF_Image.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Value_Input.H>
#include <FL/Fl_Input.H>
#include <FL/Fl_Output.H>
#include <FL/Fl_Text_Display.H>
#include <FL/Fl_Value_Output.H>
#include <FL/Fl_Multiline_Output.H>
#include <FL/Fl_PNG_Image.H>
#include <FL/Fl_JPEG_Image.H>
#include <FL/Fl_GIF_Image.H>
void cbHeight(); void animateGif(void*); void loadimages(); void displayGif(); extern std::string hiii; extern Fl_GIF_Image * gifs[];
struct BMIInfo
{
    std::string status, risk;
    double value;
};
extern BMIInfo bmi;
Fl_Cairo_Window* makeInputWindow(); extern Fl_Input * wi; extern Fl_Input * hi; extern Fl_Box * ii; extern Fl_Button * t;
Fl_Cairo_Window* makeOutWindow(); extern Fl_Cairo_Window * cw; extern Fl_Button * c; extern Fl_Cairo_Window * dw;
extern Fl_Box * gb; extern Fl_Cairo_Window * ow; extern Fl_Cairo_Window * b;
Fl_Cairo_Window* CelebWindow(); extern Fl_Output * st; extern Fl_Value_Output * bm; extern Fl_Text_Display * hr; extern Fl_Box * cb;
void cbCeleb(Fl_Button*,void*);
BMIInfo getAPIInfo(); std::string searchAPI(std::string,std::string);
const int w = 512; const int h = 600;
```

Test: Input Window

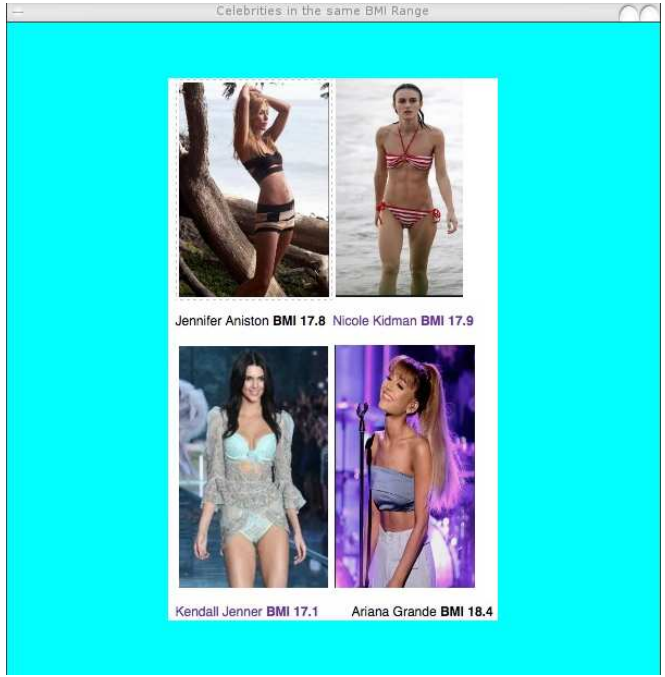
The input window with an animated gif





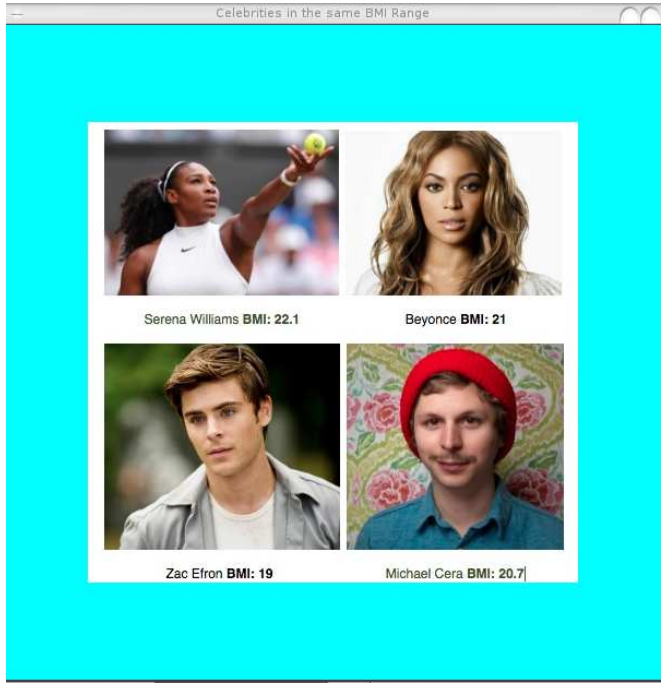
Test: Output Window (Underweight)

Output and Celeb Windows if the user is underweight. (Input 100 lbs and 64 in.)



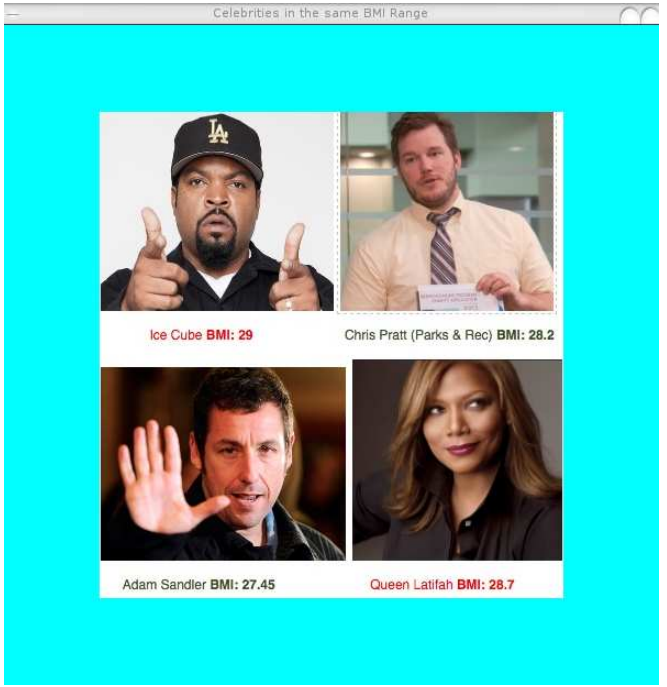
Test: Output Window (Healthy)

Output and Celeb Windows if the user is healthy. (Input 150 lbs and 66 in.)



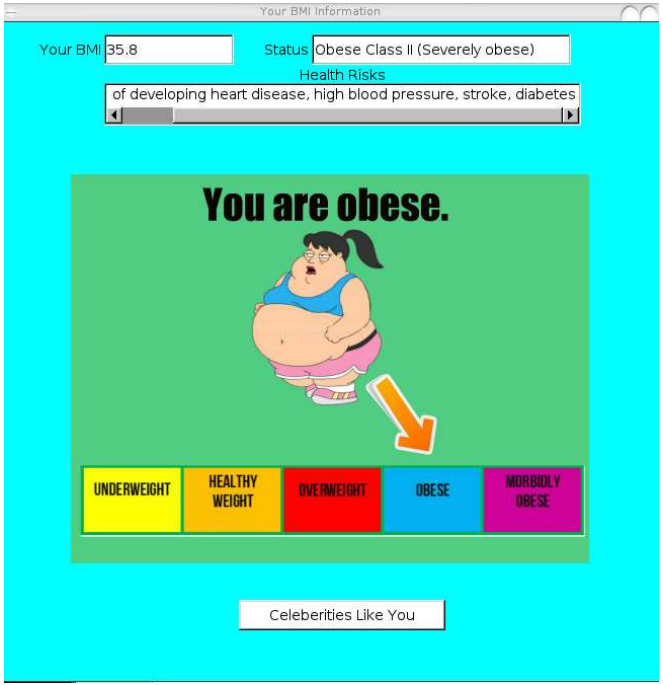
Test: Output Window (Overweight)

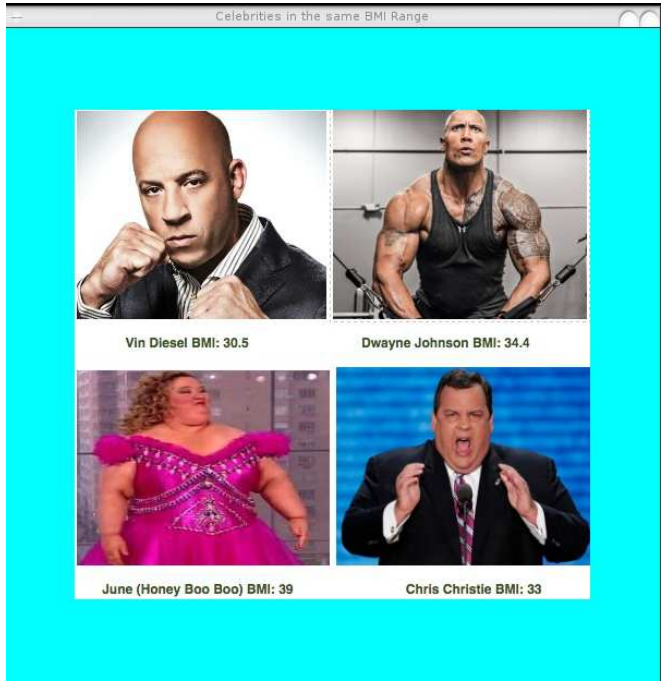
Output and Celeb Windows if the user is overweight. (Input 170 lbs and 64 in.)



Test: Output Window (Obese)

Output and Celeb Windows if the user is obese. (Input 250 lbs and 70 in.)





Test: Output Window (Extremely Obese)

Output and Celeb Windows if the user is extremely obese. (Input 250 lbs and 65 in.)