cs102 lab 3

Specification:

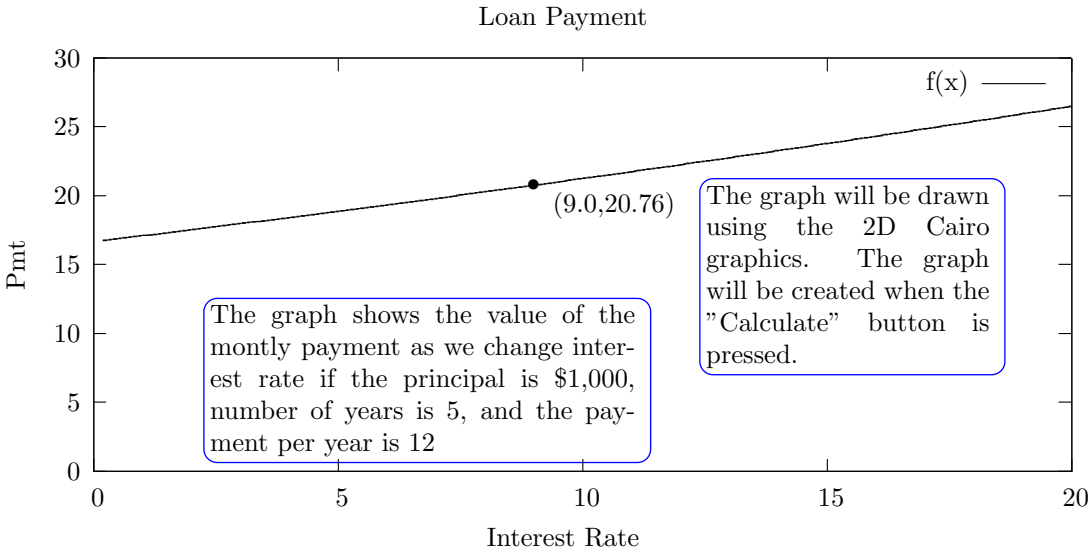
The "Payment" refers to the amount of each montly payment

$$payment = \frac{intRate \times \frac{principal}{payPerYear}}{1 - (\frac{intRate}{payPerYear} + 1)^{-payPerYear \times numYears}}$$

For example, the payment for interest rate of .09, principle of 1000, with 12 payments per year, and 5 years for loan.

$$payment = \frac{.09 \times \frac{1000}{12}}{1 - (\frac{.09}{12} + 1)^{-12 \times 5}}$$

Analysis:


Loan Payment
f(x)
(9.0,20.76)
The graph will be drawn using the 2D Cairo graphics. The graph will be created when the "Calculate" button is pressed.
The graph shows the value of the montly payment as we change interest rate if the principal is $1,000, number of years is 5, and the payment per year is 12
Pmt
Interest Rate
0
5
10
15
20
25
30

Design:

The program is split into several modules:

- lab.h
  - contains all the include files and variables shared amongst multiple cpp files
- labgui.cpp
  - declares all the FLTK Variables for the calculation GUI
- clabgui1.cpp
  - rnd: rounds off the value of the payment into cents
  - cb_calculate: programs the "Calculate" button
- clabgui2.cpp
  - the actual code just for the Calculation Window
- labgraph.cpp
  - Makes the window for the graph to go in
- cbDrawGraph.cpp
  - Creates the variables for the graph
  - Calls the functions to:
    - Draw X-Axis
    - Draw Y-Axis
    - Plot the Point

Clicking the file name will send you straight to its designated implementation page

Design (cont.):

- drawXAxis.cpp
  - Draws the X-Axis, with ticks and labels
- drawYAxis.cpp
  - Draws the Y-Axis, with ticks and labels
- plotPoint.cpp
  - Places a circle at the values of (interest rate,payment value)
- lab.cpp
  - main: Makes the main window and then contorl is move to FLTK
  - f: Calls pmt with user values for principal, payments per year, interest rate, and number of years.
  - pmt: Uses the formula to calculate and display the payment

Clicking the file name will send you straight to its designated implementation page

Implementation lab.h

List of all Variables and functions

```
#include <config.h>
#include <cmath>
#include <FL/Fl_Cairo_Window.H>
#include <FL/Fl_Value_Input.H>
#include <FL/Fl_Value_Output.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_PNG_Image.H>
#include <sstream>
void cbDrawGraph(Fl_Cairo_Window*,cairo_t*cr);
Fl_Cairo_Window * make_window();
double f(double r, double a, double ppy, double n);
double pmt(double r,double a,double ppy,double n);
Fl_Cairo_Window * make_graph();
void cb_Calculate(Fl_Button*,void*);
extern Fl_Cairo_Window * cw;
extern const int width;
extern const int height;
extern Fl_Button * b;
extern Fl_Box * g;
extern Fl_Value_Output * p;
extern Fl_Value_Input *r;
extern Fl_Value_Input *a;
extern Fl_Value_Input *ppy;
extern Fl_Value_Input *n;
```

Implementation labgui.cpp

Declarations of all FLTK variables

```
#include "lab.h"
Fl_Cairo_Window * cw;
Fl_Value_Input * r;
Fl_Value_Input * a;
Fl_Value_Input * ppy;
Fl_Value_Input * n;
Fl_Value_Output * p;
Fl_Button * b;
Fl_Box * g;
const int width = 300; //number of pixels of width of the window
const int height = 300; // same as width but for height
```

Implementation clabgui1.cpp

```
#include "lab.h"
extern Fl_Cairo_Window * cg;
double rnd(double d)
{
    d=d*100;
    d=std::round(d);
    d=d/100;
    return d;
}

void cb_Calculate(Fl_Button*,void*)
{
    p->value(rnd(f(r->value(),a->value(),ppy->value(),n->value())));
    if(cg) cg->hide();
    else cg = make_graph();
    cg->show();
}
```

Rounding works by:

- 1) Multiplying by 100
- 2) Rounding the number to the nearest whole integer
- 3) Dividing by 100 to return to dollars and cents

p's value is composition of the "rnd" function of the "f" function of the values of r, p, ppy, and n.

Implementation clabgui2.cpp

The Make Window Function on its own in it's entirety

```
#include "lab.h"
Fl_Cairo_Window * make_window(){
    cw = new Fl_Cairo_Window(width,height);
    cw->label("Lab 3:Loan Payment Calculator");
    cw->color(fl_rgb_color(121,152,182));
    a = new Fl_Value_Input(.6*width,.05*height,.25*width, .1*height);
    //number are how far in, how far down, how wide type, how tall type
    a->label("Principal:");
    r = new Fl_Value_Input(.6*width,.15*height,.25*width, .1*height);
    r->label("Interest Rate (9% = 9):");
    ppy = new Fl_Value_Input(.6*width,.25*height,.25*width, .1*height);
    ppy->label("# of Payments per Year:");
    n = new Fl_Value_Input(.6*width,.35*height,.25*width, .1*height);
    n->label("# of Year:");
    p = new Fl_Value_Output(.6*width,.75*height,.25*width, .1*height);
    p->label("Monthly Payment:");
    b = new Fl_Button(.6*width,.55*height,.25*width, .1*height);
    b->label("Calculate");
    b->color(FL_BLUE); b->labelcolor(FL_WHITE);
    b->callback((Fl_Callback*)cb_Calculate);
    g = new Fl_Box(FL_FLAT_BOX,.25*width,.535*height,64,64,"");
    g->color(fl_rgb_color(121,152,182));
    g->image(new Fl_PNG_Image("loan.png")); return cw;}
```

The name of a cairo text box is repective to the variable that the input is saved to.

Implementation labgraph.cpp

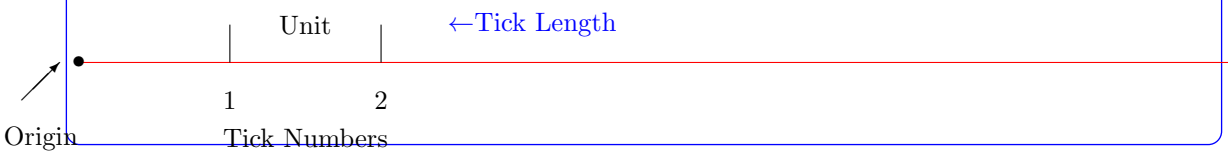
Makes a window called "cg" for the graph

```
#include "lab.h"
Fl_Cairo_Window * cg;
extern const int width;
extern const int height;
Fl_Cairo_Window * make_graph()
{
    cg = new Fl_Cairo_Window(width,height);
    cg->label("Lab 3:Loan Payment Calculator Graph");
    cg->color(fl_rgb_color(121,152,182));
    cg->set_draw_cb(cbDrawGraph);
    return cg;
}
```

Implementation cbDrawGraph.cpp

```
#include "lab.h"
void drawXAxis(cairo_t*cr,int unit,int tickLength,int ticks);
void drawYAxis(cairo_t*cr,int unit,int tickLength,int ticks);
void plotPoint(cairo_t*cr,int unit,int tickLength,int ticks);
void cbDrawGraph(Fl_Cairo_Window*,cairo_t*cr)
{
    int unit = .1*width;
    int tickLength = .5*unit;
    int ticks = width/unit;
    drawXAxis(cr,unit,tickLength,ticks);
    drawYAxis(cr,unit,tickLength,ticks);
    plotPoint(cr,unit,tickLength,ticks);
}
```

Diagram shows length of each variable made in this file



Implementation drawXAxis.cpp

```
#include "lab.h"
void drawXAxis(cairo_t* cr,int unit,int tickLength,int ticks)
{
    int x1 = unit; int y1 = height-unit;
    int x2 = width-unit; int y2 = height-unit;

    //x axis
    cairo_move_to(cr,x1,y1);
    cairo_line_to(cr,x2,y1);

    //draw tick marks
    for(int i = 2; i < 2*(ticks-1); i= i+2)
    {
        cairo_move_to(cr,x1+(i/2)*unit,y1);
        cairo_line_to(cr,x1+(i/2)*unit,y1-tickLength);
        cairo_move_to(cr,x1+(i/2)*unit,y1+tickLength);
        std::ostringstream oss; oss << i;
        cairo_show_text(cr,oss.str().c_str());
    }
    cairo_stroke(cr);
}
```

Creates a straight line to represent the X-Axis

"i" increments by 2 so the scale of the X-Axis will be 2 per tick

Implementation drawYAxis.cpp

```
#include "lab.h"
void drawYAxis(cairo_t* cr,int unit,int tickLength,int ticks)
{
    int x1 = unit; int y1 = height−unit;
    int x2 = width−unit; int y2 = unit;

    //y axis
    cairo_move_to(cr,x1,y1);
    cairo_line_to(cr,x1,y2);

    //draw tick marks
    for(int i = 5; i < 5*(ticks−1); i = i + 5)
    {
        cairo_move_to(cr,x1,y1−(i/5)*unit);
        cairo_line_to(cr,x1+tickLength,y1−(i/5)*unit);
        cairo_move_to(cr,x1−tickLength,y1−(i/5)*unit);
        std::ostringstream oss; oss << i;
        cairo_show_text(cr,oss.str().c_str());
    }
    cairo_stroke(cr);
}
```

Creates a vertical line for the Y-Axis

Creates ticks in the same way as the X-Axis but now it increments by 5 so to increase the range of the graph

Implementation plotPoint.cpp

```
#include "lab.h"
#include <iostream>
void plotPoint(cairo_t* cr,int unit,int tickLength,int ticks)
{
    int x1 = unit; int y1 = height−unit;
    const int dollarsPerUnit = 5;
    double x = r−>value(); x = x1 + x * (unit/2);
    double y = p−>value();
    y = y1 − y/dollarsPerUnit * unit;
    const double radius = 4;
    double begin = 0; double end = 2* M_PI;
    std::ostringstream oss;
    oss << "(" << r−>value() <<
    "," << p−>value() << ")";
    cairo_arc(cr,x,y,radius,begin,end);
    cairo_show_text(cr,oss.str().c_str());
    cairo_stroke(cr);
}
```

Draws a circle at the point (r value,p value) from the ”pmt” function

Implementation lab.cpp

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include "lab.h"
using namespace std;
int main ()
{

    make_window()->show();
    Fl::run();
    return 0;
}
double f(double r, double a, double ppy, double n)
{
    return pmt(r,a,ppy,n);
}
double pmt(double r, double a, double ppy, double n)
{
    return ((r/100.0) * (a/ppy)) / (1-pow((r/100.0/ppy+1),-(ppy*n)));
}
```

The function "PMT" uses 4 double variables. a=principle,r=interest rate, ppy=payments per year, n=number of years. Then it inputs each variable into the monthly payment equation returning the payment value.

$$payment = \frac{r \times \frac{a}{ppy}}{1 - (\frac{r}{ppy} + 1)^{-ppy \times n}}$$

Test

- User can enter any data they desire
- If user enters principal of $1,000, interest rate of 5%, payments per year is 12, and number of years is 5 and its point on the graph is:
- If user enters principal of $1,000, interest rate of 12%, payments per year is 6, and number of years is 10 and its point on the graph is:

The figures show the functions ability to graph any point between (0,0) to (16,40)