

Lab 3

Sloan Atkins

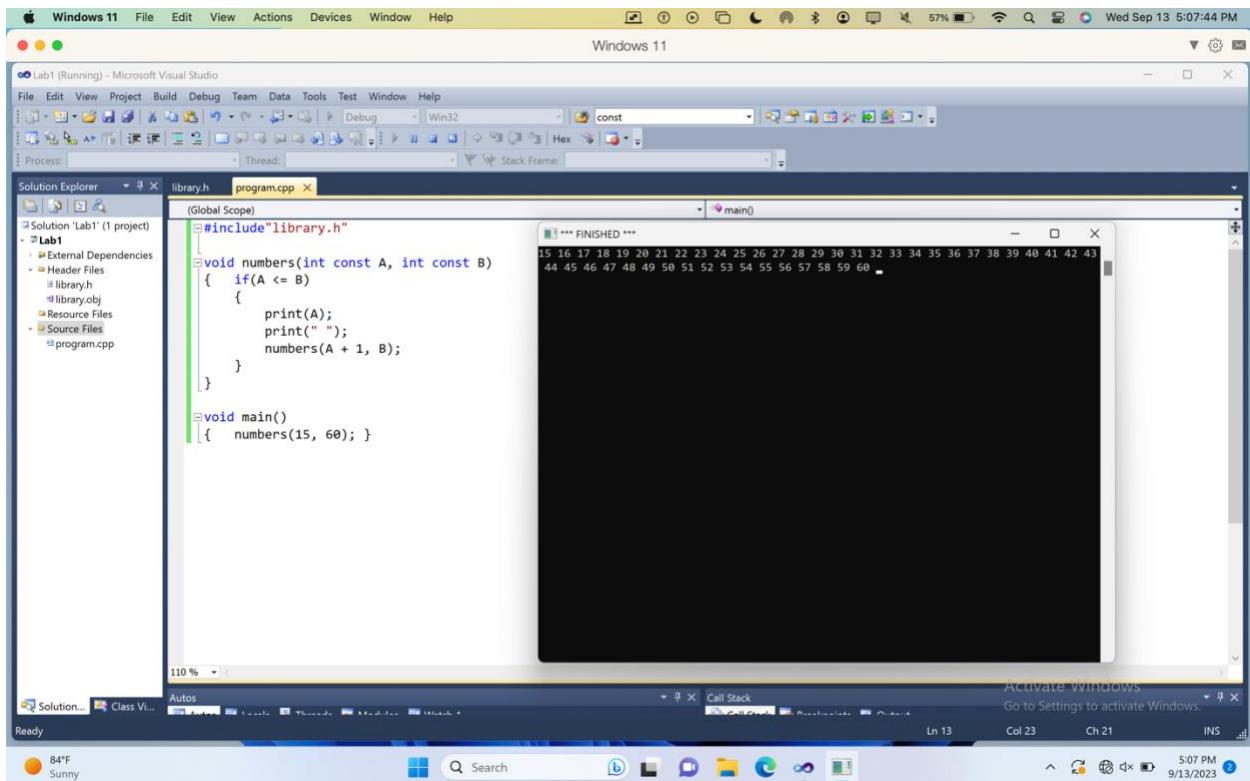
ECE 118

A1. Remembering how to start

```
#include"library.h"

void numbers(int const A, int const B)
{
    if(A <= B)
    {
        print(A);
        print(" ");
        numbers(A + 1, B);
    }
}

void main()
{   numbers(15, 60); }
```



A2. Exotic Measures

```
#include"library.h"

void numbers(int const kilograms, int const B)
{
    if(kilograms <= B)
    {
        int const pounds = (kilograms * 2.0462 + 0.5);
        int const stones = (pounds/14);
        int const stpound = (stones%14);
        print(kilograms);
        print(" kg is ");
        print(pounds);
        print(" lbs or ");
        print(stones);
        print(" st ");
        print(stpound);
        print(" lbs.");
        new_line();
        numbers(kilograms + 5, B);
    }
}

void main()
{
    numbers(20, 100); }
```

The screenshot shows the Microsoft Visual Studio interface running on a Windows 11 desktop. The code in `program.cpp` is displayed in the editor. A terminal window is open, showing the output of the program's execution. The output lists various weight conversions from kilograms to pounds and stones/stpounds for increments of 5 kg starting from 20 kg up to 100 kg. The terminal window title is "cell". The status bar at the bottom right shows the date and time as "Fri Sep 15 10:09:47 AM" and the system temperature as "84°F Sunny".

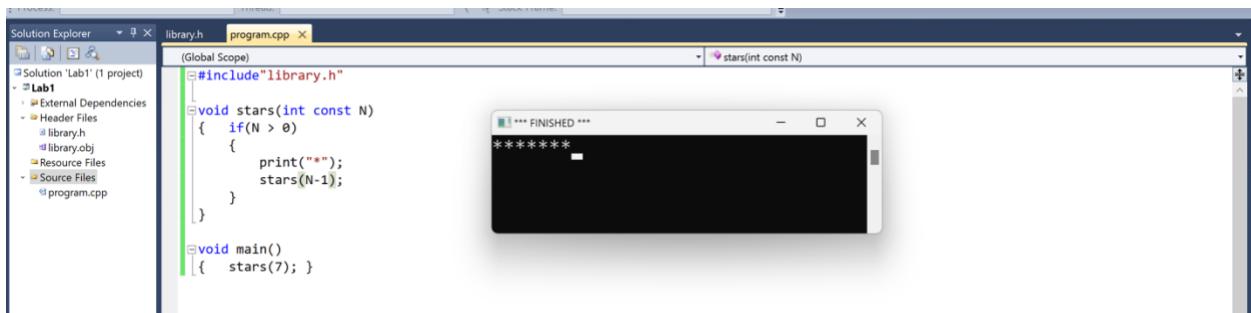
```
20 kg is 44 lbs or 3 st 3 lbs.
25 kg is 55 lbs or 3 st 3 lbs.
30 kg is 66 lbs or 4 st 4 lbs.
35 kg is 77 lbs or 5 st 5 lbs.
40 kg is 88 lbs or 6 st 6 lbs.
45 kg is 99 lbs or 7 st 7 lbs.
50 kg is 110 lbs or 7 st 7 lbs.
55 kg is 121 lbs or 8 st 8 lbs.
60 kg is 132 lbs or 9 st 9 lbs.
65 kg is 143 lbs or 10 st 10 lbs.
70 kg is 154 lbs or 11 st 11 lbs.
75 kg is 165 lbs or 11 st 11 lbs.
80 kg is 176 lbs or 12 st 12 lbs.
85 kg is 187 lbs or 13 st 13 lbs.
90 kg is 198 lbs or 14 st 0 lbs.
95 kg is 209 lbs or 14 st 0 lbs.
100 kg is 220 lbs or 15 st 1 lbs.
```

B1. Stars

```
#include"library.h"

void stars(int const N)
{
    if(N > 0)
    {
        print("*");
        stars(N-1);
    }
}

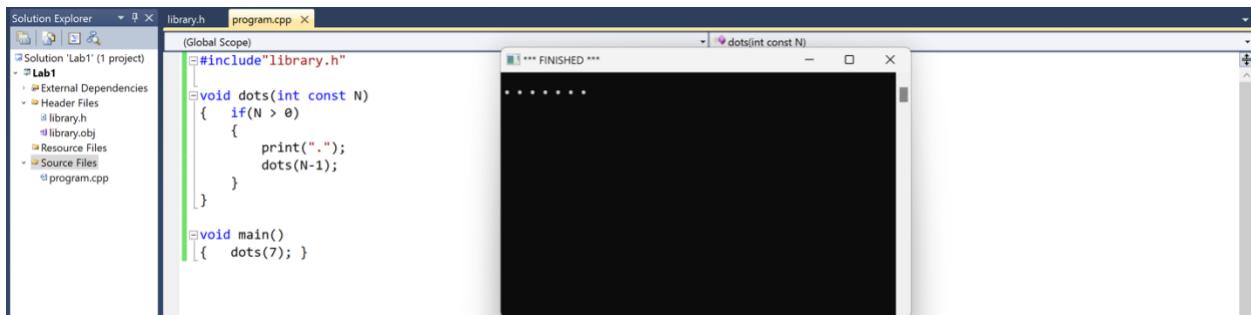
void main()
{   stars(7); }
```



B2. Dots

```
#include"library.h"
void dots(int const N)
{
    if(N > 0)
    {
        print(".");
        dots(N-1);
    }
}

void main()
{   dots(7); }
```



B3. Stars and Dots

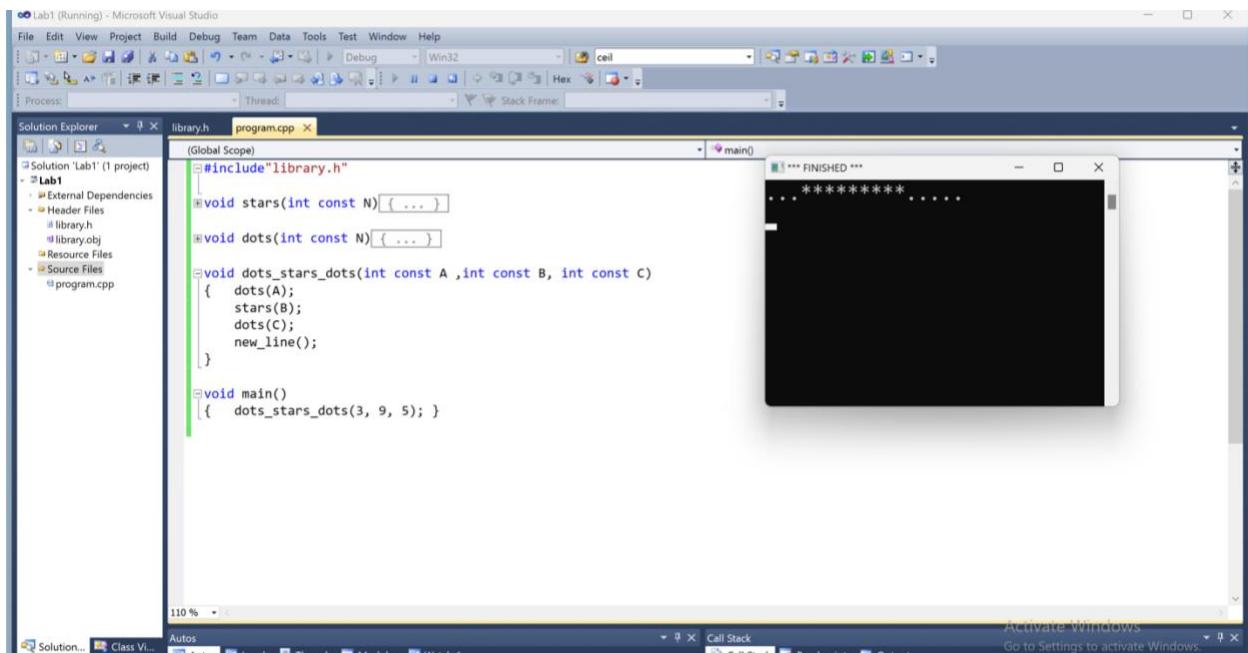
```
#include"library.h"

void stars(int const N)
{
    if(N > 0)
    {
        print("*");
        stars(N-1);
    }
}

void dots(int const N)
{
    if(N > 0)
    {
        print(".");
        dots(N-1);
    }
}

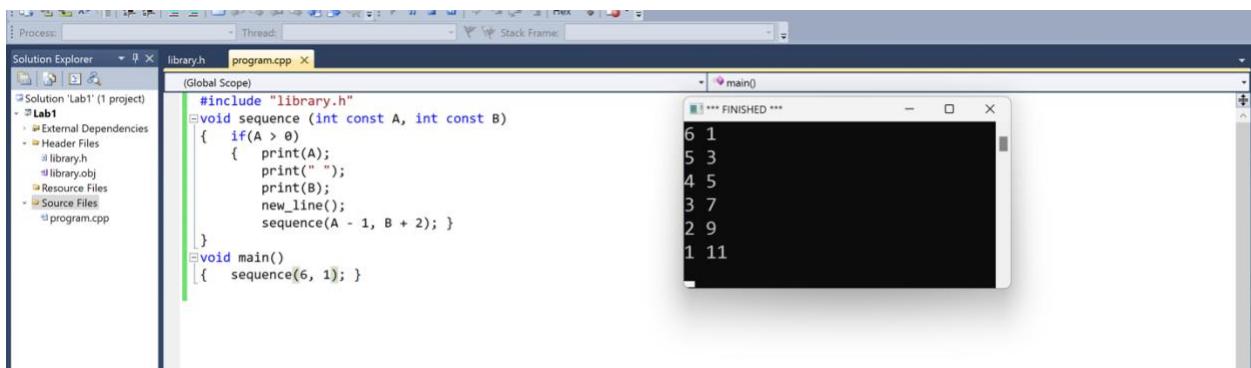
void dots_stars_dots(int const A ,int const B, int const C)
{
    dots(A);
    stars(B);
    dots(C);
    new_line();
}

void main()
{
    dots_stars_dots(3, 9, 5); }
```



B4. Another adaptation

```
#include "library.h"
void sequence (int const A, int const B)
{   if(A > 0)
    {   print(A);
        print(" ");
        print(B);
        new_line();
        sequence(A - 1, B + 2); }
}
void main()
{   sequence(6, 1); }
```



B5. Combining

```
#include"library.h"

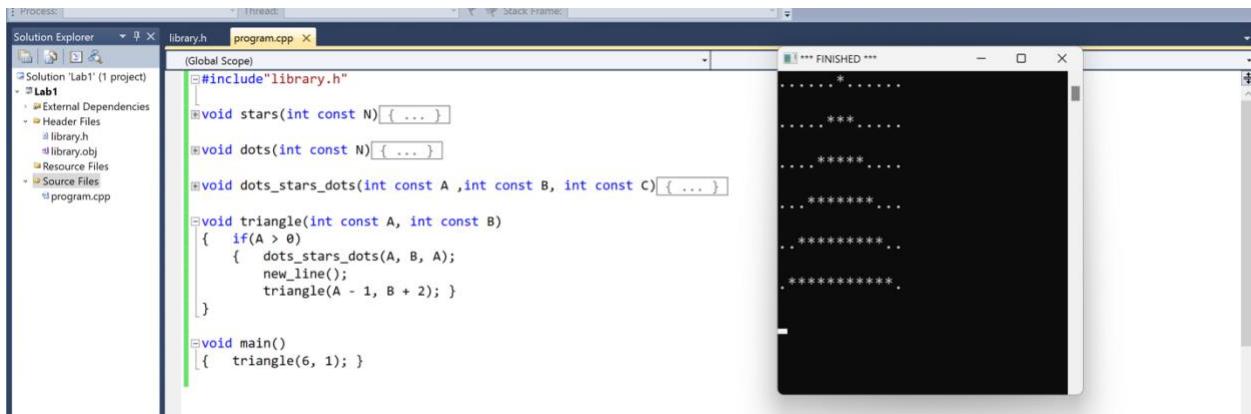
void stars(int const N)
{
    if(N > 0)
    {
        print("*");
        stars(N-1);
    }
}

void dots(int const N)
{
    if(N > 0)
    {
        print(".");
        dots(N-1);
    }
}

void dots_stars_dots(int const A ,int const B, int const C)
{
    dots(A);
    stars(B);
    dots(C);
    new_line();
}

void triangle(int const A, int const B)
{
    if(A > 0)
    {
        dots_stars_dots(A, B, A);
        new_line();
        triangle(A - 1, B + 2); }
}

void main()
{   triangle(6, 1); }
```



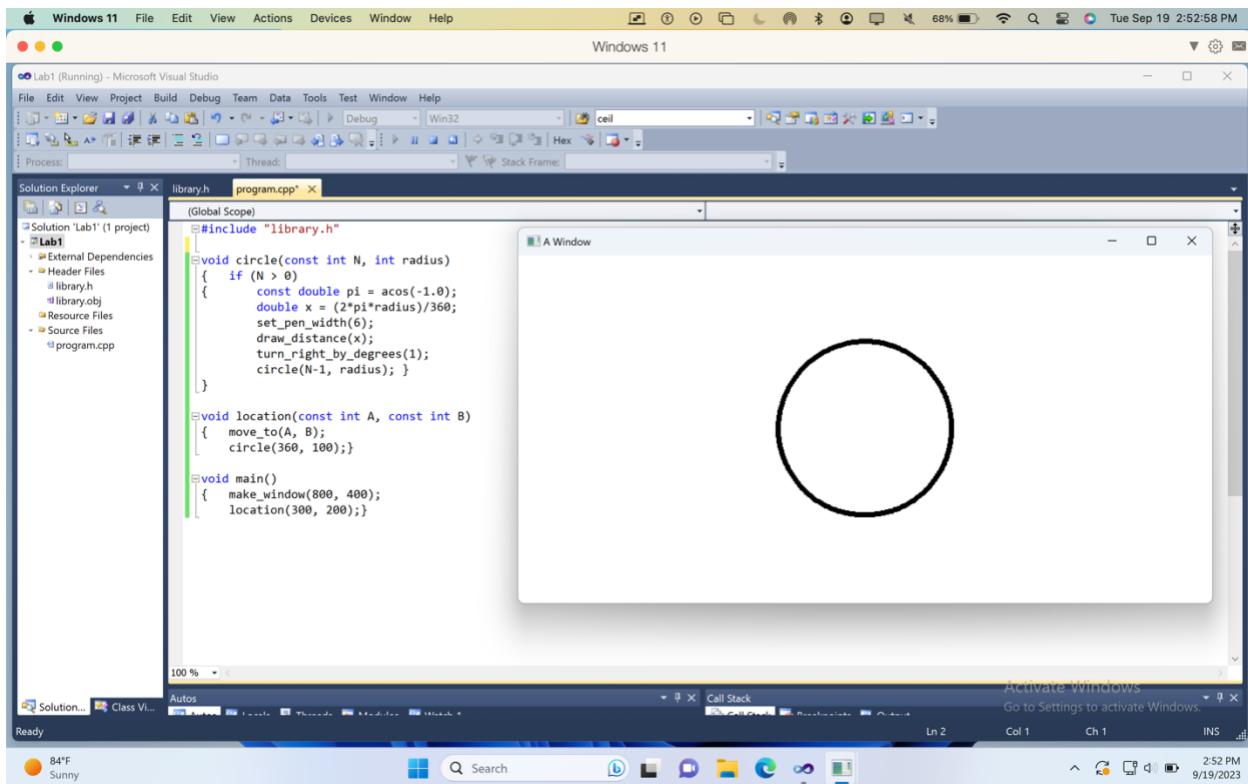
C1. A circle

```
#include "library.h"

void circle(const int N, int radius)
{
    if (N > 0)
    {
        const double pi = acos(-1.0);
        double x = (2*pi*radius)/360;
        set_pen_width(6);
        draw_distance(x);
        turn_right_by_degrees(1);
        circle(N-1, radius); }
}

void location(const int A, const int B)
{
    move_to(A, B);
    circle(360, 100);}

void main()
{
    make_window(800, 400);
    location(300, 200);}
```



C2. Weaponizing

```
#include "library.h"
void circle(const int N, int radius, int degrees)
{
    if (N > 90 - degrees)
    {
        const double pi = acos(-1.0);
        double x = (2*pi*radius)/360;
        set_pen_width(6);
        draw_distance(x);
        turn_right_by_degrees(1);
        circle(N-1, radius, degrees);}
}

void draw_cannon(const double radius)
{
    set_pen_color(color::black);
    set_pen_width(5);
    start_shape();
    turn_left_by_degrees(90);
    draw_distance(0.5*radius);
    turn_right_by_degrees(90);
    draw_distance(1.25*radius);
    turn_right_by_degrees(94);
    draw_distance(4*radius);
    turn_right_by_degrees(87);
    draw_distance(0.85*radius);
    turn_right_by_degrees(87);
    draw_distance(2.68*radius);
}

void location(const int A, const int B)
{
    move_to(A, B);
    circle(720, 40, 50);
    draw_cannon(100);}

void main()
{
    make_window(800, 400);
    location(300, 300);}
```

