# Coding Assignment 5

Due to time and effort constraints, we are taking some "shortcuts" in our GTKMM coding. Our goal is to exercise a small subset of the GTKMM functionality. For now, we are ignoring GTKMM warnings that occur during run time as long as the proper results are obtained.

Given that, we will be adding our GTKMM code directly into Code5_xxxxxxxxxx.cpp (referred to as just Code5.cpp in the rest of this document).

The descriptions of the necessary changes in the rest of this document make some assumptions about how the code is written. If your code is written differently, you will need to adapt the changes to fit your code.

This document will refer to the "Machine List Menu"

```
Pick a Coke Machine



0.  Exit
1.  Machine Bugs Bunny
2.  Machine Cecil Turtle
3.  Machine Daffy Duck
4.  Machine Elmer Fudd
5.  Machine Fog Horn
6.  Add a new machine

Enter choice
```

and this document will refer to the "Coke Menu"

```
0. Walk away
1. Buy a Coke
2. Restock Machine
3. Add change
4. Display Machine Info
5. Update Machine Name
6. Update Coke Price
```

For this assignment, the Machine List Menu will not change. The Coke Menu will be replaced with GTKMM functions.

If your Coke Menu is not already a separate function from `main()`, you should make that change and test it BEFORE starting on this assignment.

Changes in `main()`

Add this line at the beginning of `main()`

```
Gtk::Main kit(argc, argv);
```

The `case` values in the `switch` statement in `main()` correspond to the menu values in Coke Menu.

Case 1 – Buy a Coke

Create a new function to get the payment. The function takes no parameters and returns the `int` value into the payment variable used in case 1's code. See description of the GTKMM code later in this document

Based on the action returned by `buyACoke()`, the code currently prints different message. All of those text based messages should be converted to MessageDialog boxes.

The code for the MessageDialog should consist of 5 lines

Form the message being sent to the MessageDialog

Create the MessageDialog using `new`

Run it

Close it

Delete it

Case 2 – Restock Machine

Create a new function to get the restock amount. The function takes no parameters and returns the `int` value into the amount of inventory to add variable used in case 2's code. See description of the GTKMM code later in this document.

Based on the result of restocking the machine, the code either prints that the restock was successful or that the request exceeded capacity. Both of the text based messages should be converted to MessageDialog boxes. Use the same 5 lines of code described in Case 1.

Case 3 – Add change

Create a new function to get the change amount to add. The function takes no parameters and returns the `int` value into the amount of change to add variable used in case 3's code. See description of the GTKMM code later in this document.

Based on the result of adding change to the machine, the code either prints that the addition was successful or that the request exceeded capacity. Both of the text based messages should be converted to MessageDialog boxes. Use the same 5 lines of code described in Case 1.

Case 4 – Display Machine Info

Direct your overloaded << output into a new ostringstream variable and then pull the string out of ostringstream variable and display it using a MessageDialog box. Use the same 5 lines of code.

Case 5 – Update Machine Name

After calling your existing `getMachineName()`, use a MessageDialog box to display the message that the machine has been updated (same 5 lines of code).

Case 6 – Update Coke Price

After calling your existing `setCokePrice()`, use a MessageDialog box to display the message that the Coke price has been updated (same 5 lines of code).

In your function that prints Coke Menu, replace the text based menu with a dialog with buttons for the choices.

Create a dialog

Add buttons to the dialog for each menu option

Run the dialog and store the result returned by the run.

Use that result to set the same variable that your original text menu set. Your `main()` should be expecting the Coke Menu function to return what choice you picked from the text menu- now you will be returning the same choice value just from the dialog box.

**Function to get payment**

Create a dialog box

Create an entry box and manage it

Add buttons to the dialog box

Run the dialog and capture the return value into a variable

Close and delete the dialog box

Return the variable

**Function to get restock value**

Copy code used to get payment and alter for restock

**Function to get change to add**

Copy code used to get payment and alter for adding change

**Function to get new Coke price**

Copy code used to get payment and alter for new Coke price

**Function to get new machine name**

Copy code used to get payment and alter for get new machine name

This sample code was discussed in class. I am including it here for your reference.

```
// Dialog Example
#include <iostream>
#include <gtkmm.h>

int main (int argc, char* argv[])
{
      Gtk::Main kit(argc, argv);

      Gtk::Dialog *MyDialog = new Gtk::Dialog();

      MyDialog->set_title("Question");

      std::vector<std::string> buttons = {"Cancel", "OK", "Maybe", "Maybe Not?", "Who Cares"};
```

```
      for (int i = 0; i < buttons.size(); i++)
            MyDialog->add_button(buttons[i], i);
      MyDialog->set_default_response(4);

      Gtk::Label *MyLabel = Gtk::manage(new Gtk::Label("What is your decision?"));
      MyDialog->get_vbox()->pack_start(*MyLabel);
      MyLabel->show();

      Gtk::Entry *MyEntry = Gtk::manage(new Gtk::Entry());
      MyEntry->set_text("Enter your name here");
      MyEntry->set_max_length(50);
      MyEntry->show();
      MyDialog->get_vbox()->pack_start(*MyEntry);

      Gtk::Image *MyImage = Gtk::manage(new Gtk::Image{"gtkmm-SMALL.png"});
      MyImage->show();
      MyDialog->get_vbox()->pack_start(*MyImage);

      int result = MyDialog->run();
      std::string text = MyEntry->get_text();

      MyDialog->close();

      delete MyDialog;

      std::cout << "User " << text << " clicked \"" << buttons[result] << "\"" << std::endl;

      return 0;
}

-----------------------------------------------------------------------------------------------
// MessageDialog Example
#include <gtkmm.h>

int main(int argc, char *argv[])
{
      Gtk::Main kit(argc, argv);

      std::string message{"This is my message"};

      Gtk::MessageDialog *MyMessageDialog  =
      new Gtk::MessageDialog{message, true,Gtk::MESSAGE_INFO, Gtk::BUTTONS_OK, false};
```

```
    MyMessageDialog->run();
    MyMessageDialog->close();
    delete MyMessageDialog;
}
```