

# CSC 3110 Program # 2: Pointing at Palindromes

Due: October 30, by 11:59 PM via Canvas



## Program Instructional Guide

### - Intro

The term palindrome is defined as a word, phrase, number, or other sequence of characters which reads the same backward as forward, such as MADAM or KAYAK. The term “palindrome” has Greek roots from the words palin (“again”) and dromos (“way, direction”) by way of the English writer Ben Jonson. Some palindromes seem to be philosophical. Others tell a story. In any case, they are unique and often times creatively thoughtful compositions of literature used for constrained writing.

For this program, **you will write a program that utilizes pointers for palindrome testing.** The program is a simulated interaction of an airline account creation. The account parameters will consist of a passenger name, password and pin. Once the account has been created the program will test if the passenger’s password is a palindrome and print the result out to the screen (i.e, palindrome, not a palindrome).

## Program Objectives

The goal of this project is to create a C++ program that will help to develop your programming skills in understanding the implementation of specifically **pointers**, in addition to developing void functions, structures and conditionals. The simulated airline account creation will initially ask the user to enter a *name*, *birthdate*, *password* and *pin*. You should have a welcome introduction to your airline ticket purchaser notifying them of the name of your commercial airline.

### - Palindrome Function

Your program will then create a function (**check\_Palindrome**) that will test whether the passenger’s password is a palindrome or not. If the password entered is a palindrome, the palindrome string will be printed to the screen with a prompt notifying the user of the password palindrome status. Similarly, if the password is not a palindrome a print statement notifying the user that the string is not a palindrome is

given. The program should include pointers within the function that can reference, return and track the address of the string value that is being checked as a palindrome.

Hint: Declaration of pointers (i.e., datatype char is not required for implementation but serves as an example only; type can be string, etc...

```
char *c = str;
```

```
char **p = &c; // pointer to a pointer
```

The address of the string will then be printed to the screen.

You must adhere to good programming techniques and style while making sure your code is efficient. You should spend a significant amount of time thinking about what chunks of code will be reused and how you can take advantage of this when you write your program.

**Program Requirements** (*Note: Students program must follow these requirements, points will be deducted if your program fails to implement all of the requirements listed.*)

Here is a checklist of things that your program should include:

- Create a passenger account by entering a **name, password, birthdate and pin**, use the account information in the program.
- Validate the user input (For example, make sure that the passenger account input entered corresponds to the datatype declared initially).
- Your program must use pointers in both the palindrome and permutation functions.
- Your program must have a function similar to the (**check\_Palindrome**) that will test whether the passenger's password is a palindrome or not.
- Your program must implement a function similar to **rotate\_Positions** shown above that should be used to change and also temporary store the rotation of each character in the string.
- Give appropriate feedback so it's clear to the customer how the program is progressing and what is the next step or input needed to complete the account creation.
- Provide a short introduction to your program and a conclusion.

- **Your program must use at least one (1) functions: `check_Palindrome ( )`, (1) conditional (`if/if else/elif`) statements, (1) `for` and/or `while` loops and the use of pointers as defined above in the program objectives.**
- Your main function should not do all the work; it should call other functions to run the "guts" of the program.
- You must use comments in your source code to explain the flow of the program.

## Programming Language

You will write your source code using the C++ programming language. You can develop your code in the Visual Studio IDE or Mac Xcode. **However, your completed code will need to be in the format of a .cpp file extension and can run in Mac Xcode compiler for testing and grading purposes.**

## Coding conventions

- Use good modular design. Think carefully about the functions and data structures (variables) that you are creating before you start writing code.
- The main function should not contain low-level details. In other words, the main function should be a high-level overview of your solution, with the low-level details hidden in functions. This is the idea of *abstraction*.
- As a general guide, no function should be longer than a page of code. Of course there are exceptions, but these should truly be the exception.
- Use good error detection and handling. Always check return values from functions, and handle errors appropriately.
- Use descriptive names for variables and functions. You don't want to make function and variable names too long, but they should be descriptive (e.g. use "getRadius" or "get\_radius" rather than "foo" for a function that returns the value of the radius of a circle).
- Pick a capitalization style for function names and variable names, and stick with it. For example, for function name style you could do something like "square\_the\_biggest" or "squareTheBiggest" or "SquareTheBiggest".  
Programming convention is to start with a lowercase name and capitalize the second with underscores separating the words.
- Try to keep the length of each line manageable.
- Comment your code! Any complicated, tricky, or ugly code sequences in a function body should contain comments nearby describing what is going on (here is where using good function and variable names can save you from having to add comments). These comments are important to use in complicated parts of your code, but it is important to not go nuts here; over-commenting

your code can be as bad as under-commenting it. Avoid commenting the obvious. Your choice of good function and variable names should make much of your code readable.

## • **Program- and function-level comments**

These guidelines apply to this program and all future programs.

- **You should place a comment section at the top of your program that contains:**
  - Your name, the date, and the class.
  - The honor pledge saying you have neither given nor received any help you weren't supposed to.
  - A description of the program (a few sentences).
  - A description of the input to the program (what the user types), if any.
  - A description of the output of the program (what the user sees).

Here's an example:

```
/* Name: Tisha Brown
# Date: 10/29/2018
# Class: CSC 1120
# Pledge: I have neither given nor received unauthorized aid on this
program.
# Description: RPSLS is a program that allows users to play the
original RPS but now with cooler gestures.
# Input: The user types commands such as ..... (explain
commands).
# Output: The program will display the winner after several rounds of
play.
*/
```

- **For every function you write (except `main`), you must include a comment immediately before the function definition line that contains:**
  - A description of what the function does.
  - Any parameters the function takes (if any), along with their meanings.
  - The return value of the function, if there is one, and what it means.

## About the pledge

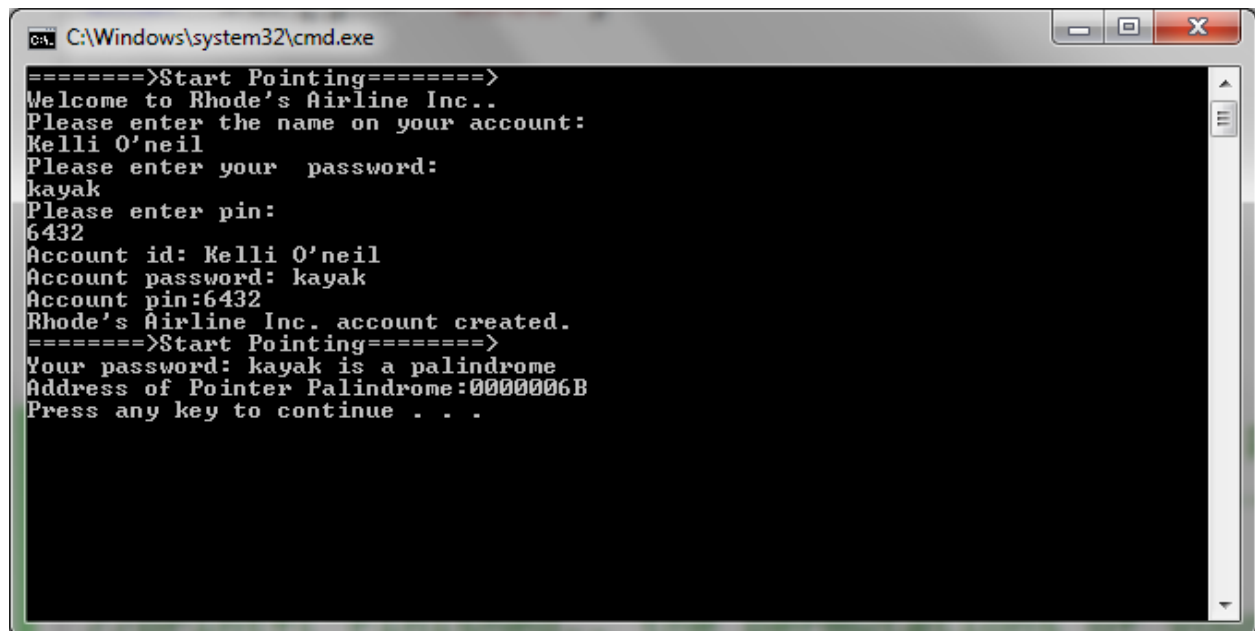
Because we are writing complex programs, it is not uncommon to get stuck. However, as the syllabus states, all homework assignments you complete outside of class must be entirely your own work, with the exception of help from the instructor, tutors, or

other people **while respecting the "Rules for Completing Assignments Independently"** (see the syllabus).

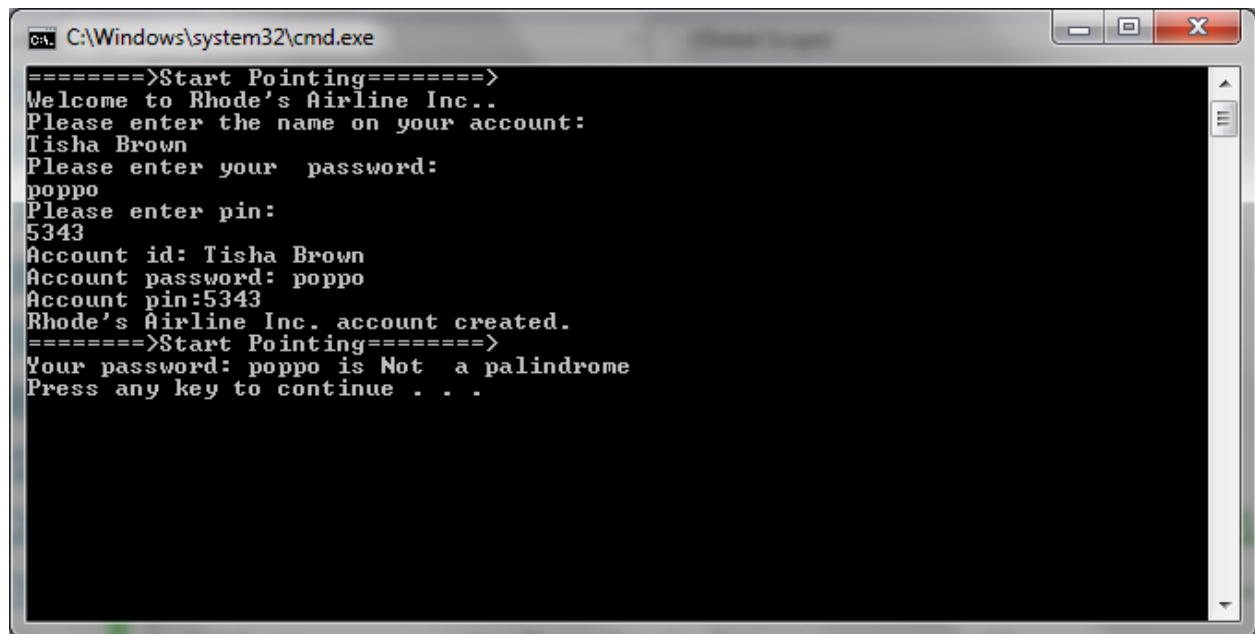
## Testing your program

You should test your program thoroughly to make sure it works. Your program should gracefully handle situations where numbers or strings are typed incorrectly for the selection option of the initial menu.

### Sample interaction (Successful run ... password is a palindrome)



```
C:\Windows\system32\cmd.exe
=====>Start Pointing=====>
Welcome to Rhode's Airline Inc..
Please enter the name on your account:
Kelli O'neil
Please enter your password:
kayak
Please enter pin:
6432
Account id: Kelli O'neil
Account password: kayak
Account pin:6432
Rhode's Airline Inc. account created.
=====>Start Pointing=====>
Your password: kayak is a palindrome
Address of Pointer Palindrome:0000006B
Press any key to continue . . .
```



```
C:\Windows\system32\cmd.exe

=====>Start Pointing=====>
Welcome to Rhode's Airline Inc..
Please enter the name on your account:
Tisha Brown
Please enter your password:
poppo
Please enter pin:
5343
Account id: Tisha Brown
Account password: poppo
Account pin:5343
Rhode's Airline Inc. account created.
=====>Start Pointing=====>
Your password: poppo is Not a palindrome
Press any key to continue . . .
```

**Sample interaction (Successful run ... password is not a palindrome)**

## What to turn in

Through Blackboard, turn in your code as a file called `Pointers_yourFirstInitial_yourLastName.cpp`.

Note: You must create pseudocode to conceptualize the flow and sequence of order for each major function within this program. The pseudocode will help you to think about the order and flow of the data within the code. Also, you must have your pseudocode available for review if you require assistance or have questions about the program (i.e., during office hours).

Questions:

Email me at [tisha.gaines@belmont.edu](mailto:tisha.gaines@belmont.edu)