\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC253 C# ProGRAMMING**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LAB 02 **CONTROL STRUCTURES**

# Objectives

In this lab assignment, students will learn:

* How to create and use counter-controlled loops
* How to create and use sentinel-controlled loops
* How to write and use if…else statements
* How to write and use switch statements

# Goals

In this lab assignment, students will demonstrate the abilities to:

* Create and use counter-controlled loops
* Create and use sentinel-controlled loops
* Write and use if…else statements
* Write and use switch statements

# Description

Create a C# console application for each question. When you create a new C# project, Visual Studio creates a folder to hold every file and sub-folder for your project. You need to zip this folder and submit the zip file to Blackboard.

1. A restaurant has 4 lunch combos for customers to choose:

Combo 1: Fried chicken with slaw [price: 4.25]

Combo 2: roast beef with mashed potato [price: 5.75]

Combo 3: Fish and chips [price:5.25]

Combo 4: soup and salad [price: 3.75]

Write a program for a group of customers to place orders. Display the menu. Ask user to enter the size of the group. Use a counter-controlled loop to take the order for each customer. Each customer should enter a combo number from 1 to 4. If an invalid choice is entered, ignore the order of that customer. At the end, count and display the number of orders for combo 1, 2, 3 and 4. Also calculate and display the total amount due from the whole group.

Example:

Today's lunch menu:

Lunch Combo 1: Fried chicken with slaw [price: 4.25]

Lunch Combo 2: roast beef with mashed potato [price: 5.75]

Lunch Combo 3: Fish and chips [price:5.25]

Lunch Combo 4: soup and salad [price: 3.75]

How many people in the group? 5

Choice of customer #1 [1/2/3/4]: 2

Choice of customer #2 [1/2/3/4]: 0

Choice of customer #3 [1/2/3/4]: 1

Choice of customer #4 [1/2/3/4]: 4

Choice of customer #5 [1/2/3/4]: 2

Order Summary:

Lunch Combo 1: 1

Lunch Combo 2: 2

Lunch Combo 3: 0

Lunch Combo 4: 1

Total amount due: $19.50

1. Write a program for course registration. This program adds/drops students and changes the capacity of a course. Set the capacity of the course to 2 initially. Keep track of number of seats taken and number of students on waiting list. Create a loop and ask the user to enter 1 to add a student, 2 to drop a student, 3 to change capacity or 0 to exit. If “add” is chosen, increase number of seats taken by one if the course is not full. Otherwise, increase number of students on waiting list by one. Display a message to tell the user what action is taken. If “drop” is chosen, display a message stating the class is empty if nobody is registered. Otherwise, decrease number of seats taken by one and display a message. If the waiting list is not empty, fill the empty space with a student currently on waiting list and display a message. If “change capacity” is chosen, ask the user to enter new capacity. If it is negative, display error message and ignore the request. If new capacity is smaller than number of seats taken currently, display error message and ignore the request. Otherwise, set the new capacity. If the class was full and new seats become available, register as many students who are on waiting as possible and display a message. Every time an “add”, “drop” or “change capacity” operation is finished, display class capacity, number of seats taken and number of students on waiting list.

Example:

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 2

Class already empty. Drop is not allowed.

Capacity: 2 Seated taken: 0 Waiting list: 0

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 1

One student added.

Capacity: 2 Seated taken: 1 Waiting list: 0

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 2

One student dropped

Capacity: 2 Seated taken: 0 Waiting list: 0

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 1

One student added.

Capacity: 2 Seated taken: 1 Waiting list: 0

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 1

One student added.

Capacity: 2 Seated taken: 2 Waiting list: 0

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 1

Class already full. Student added to waiting list.

Capacity: 2 Seated taken: 2 Waiting list: 1

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 1

Class already full. Student added to waiting list.

Capacity: 2 Seated taken: 2 Waiting list: 2

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 1

Class already full. Student added to waiting list.

Capacity: 2 Seated taken: 2 Waiting list: 3

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 1

Class already full. Student added to waiting list.

Capacity: 2 Seated taken: 2 Waiting list: 4

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 3

Enter new capacity: -1

New capacity cannot be negative.

Capacity: 2 Seated taken: 2 Waiting list: 4

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 3

Enter new capacity: 1

New capacity cannot be lower than seats taken.

Capacity: 2 Seated taken: 2 Waiting list: 4

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 3

Enter new capacity: 5

Capacity changed.

3 waitlisted students added to class.

Capacity: 5 Seated taken: 5 Waiting list: 1

Enter 1 to add, 2 to drop, 3 to change capacity, 0 to exit: 0

# Grading rubric

Program 1:

Creating loop [12.5 pts]

Taking orders [12.5 pts]

Combo counts [12.5 pts]

Total amount due [12.5 pts]

Program 2:

Creating loop [12.5 pts]

Add student [12.5 pts]

Drop student [12.5 pts]

Change capacity [12.5 pts]