CIS 2542

FALL SEMESTER 2019

PROFESSOR KATHERINE PAPADEMAS

MIKE APREZA

December 12th, 2019

Updated: January 6, 2020

THE NORTHWOODS AIRPORT MANAGEMENT SYSTEM

TABLE OF CONTENTS

INTRODUCTION

I did some research about Airport Management Systems (AMS) and came upon solid points about the foundation of an AMS. First, the system is broken up into departments. However, since the system the client needs is small, there will be no need for the main menu to be broken into departments. Also, the management system must be extremely user friendly considering how large they can become. The point that is the driving force of this AMS is that almost each function must be separated into menus. For example, if the user wishes to manipulate existing data that exists in a department called the "Airplane Department," then the menu that follows that should lead to the user having to choose which sub-department s/he wishes to go into. After that, the menu becomes more specific until exact action can be executed based on the user's desires ("Airport Technology Management: Operations, Software Solutions and Vendors", 2018).

When in the process of designing this system, I had in mind of the user being a new employee who has no clue of what is happening, which is why I decided to add to the main menu an option that gives the user information as to what the other options in the main menu lead to.

The system the client needs is one that driven heavily be menus. Some of the data that the user interacts with needs to be held throughout the whole run of the program, so I decided to user pointers for such data. For example, when it comes to accessing employee information, the user should not have to reenter the data every time they decide to use that function. Therefore, I have decided to declare a pointer to a vector of structs, which is set equal to the address of a vector of structs that is declared before the pointer. Whenever the user wishes to call said function, the pointer will be sent to it and the user can manipulate the data and return to the main menu without having to be concerned about the erasure of the records. A lot of the functions will work that way. Some functions will use static data and have no need to the previous technique.

After the user decides what they wish to do from the main menu, they should be taken to another menu to decide which action they wish to execute. From, there the appropriate actions are executed.

CLIENT INFORMATION

Client Contact Name: Mr. Nick Nichols, President
Client Company: Northwoods Airport, Inc.
Company Address
425 Fawell Blvd
Glen Ellyn, IL 60137
Company Telephone number: 630-942-2800

SCOPE OF PROJECT

The system will allow the authorized airport employee to choose from a menu to (a) add or remove airplanes waiting to land or depart from a runway; (b) enter and view employee information; (c) add customer baggage in appropriate containers per flight; (d) determine how many planes are waiting to land; (e) determine number of customers waiting for security clearance; and (f) determine the total distance a flight takes from a specific destination hub to a specific arrival hub.

PROJECT REQUIREMENTS

The Northwoods Airport, Inc., wishes to have an airport management system. The program for this system is to be written in C++ and will allow their authorized employees to access the system via the appropriate choices as specified in the scope of the project.

The client expects the application to be without errors, show inheritance, arrays and other appropriate data structures, functions, selective and iterative control structures, etc.

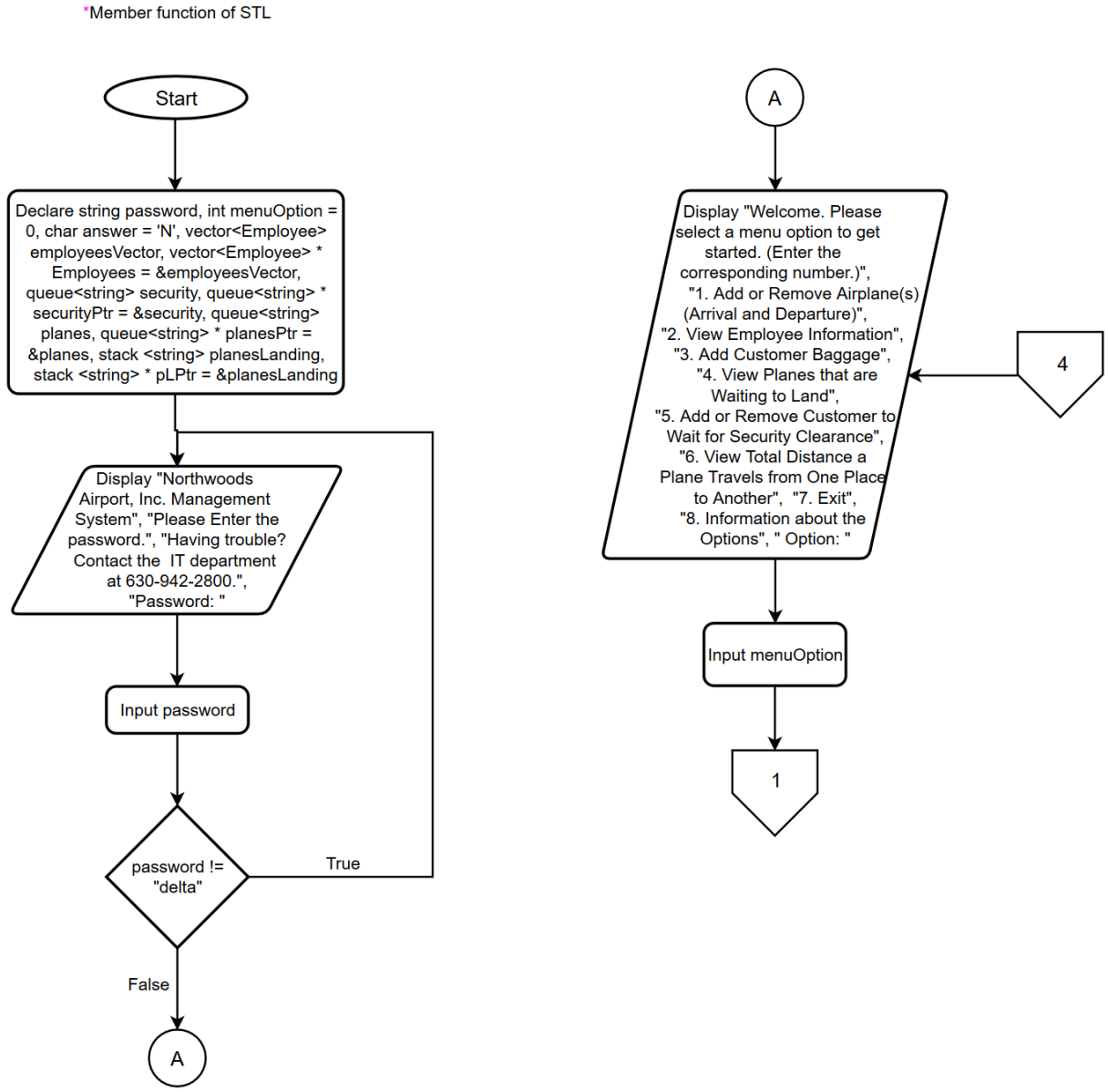
To help their programmer with the project, the client has offered the following:

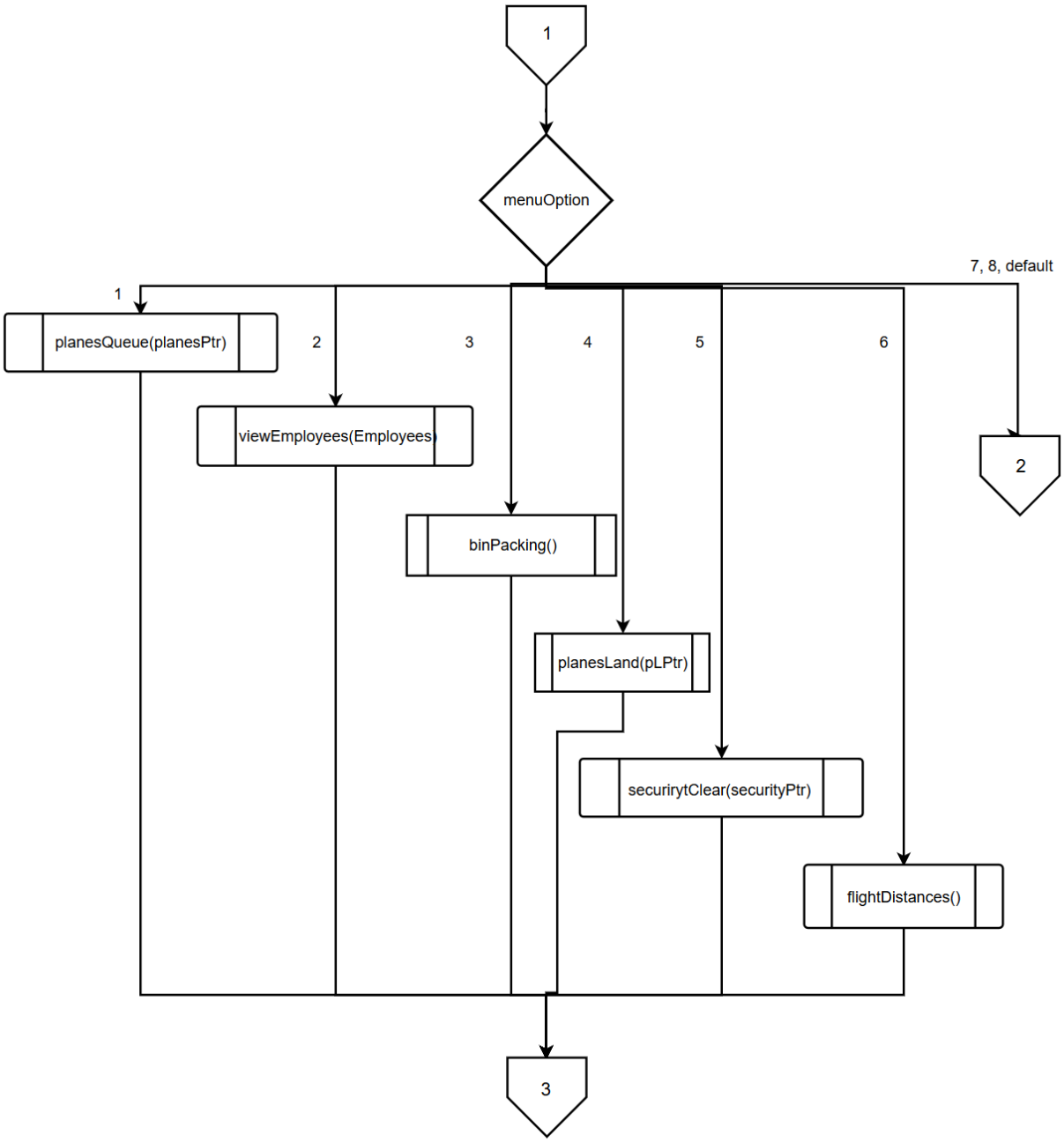
1) Construct a logon function for the employee; the password is delta (all lowercase);
2) Construct for the employee the menu to choose the function/use;
3) After the employee has made the choice, display to the employee their menu choice.
4) Once this menu has been constructed and tested, add functionality to each of the choices in the menu; for example, if the employee chose to view the employee information, then a struct and an array should be part of the process; or if the employee wants to see the airplanes waiting to land, then create a stack, with all pertinent menu items.
5) To further help the programmer, the client will allow only **three menu items** to have functionality; the remaining three items will be considered as a bonus for the programmer!

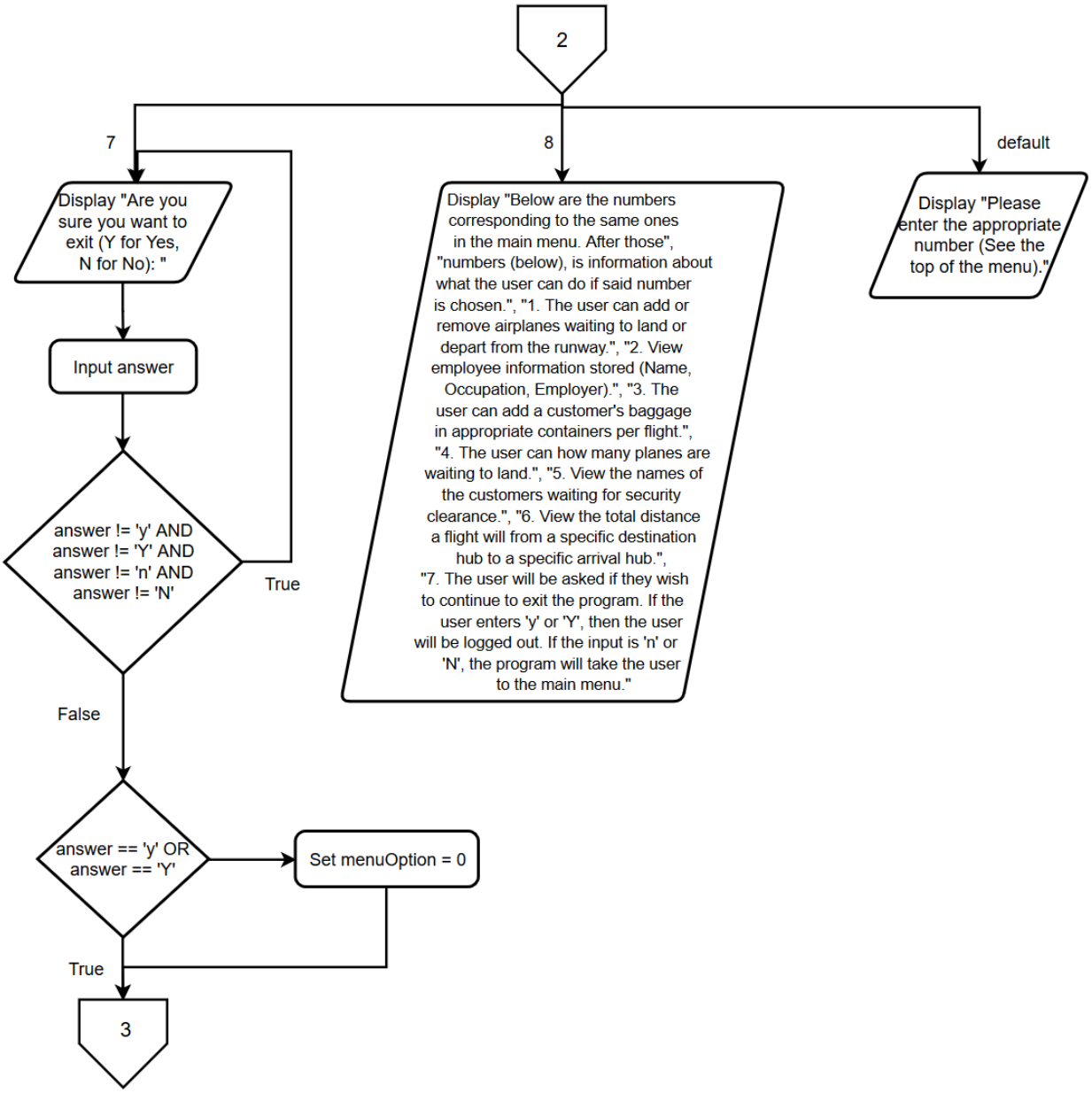
The client must have this application submitted by 12/12/2019 Thursday by midnight, along with this report (as modified). The client also wants the C++ file(s) associated with this application and will also be the owner of the software for the system.

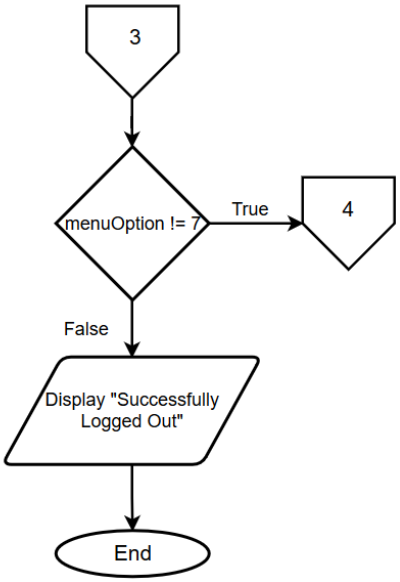
Flow Chart for Entire System

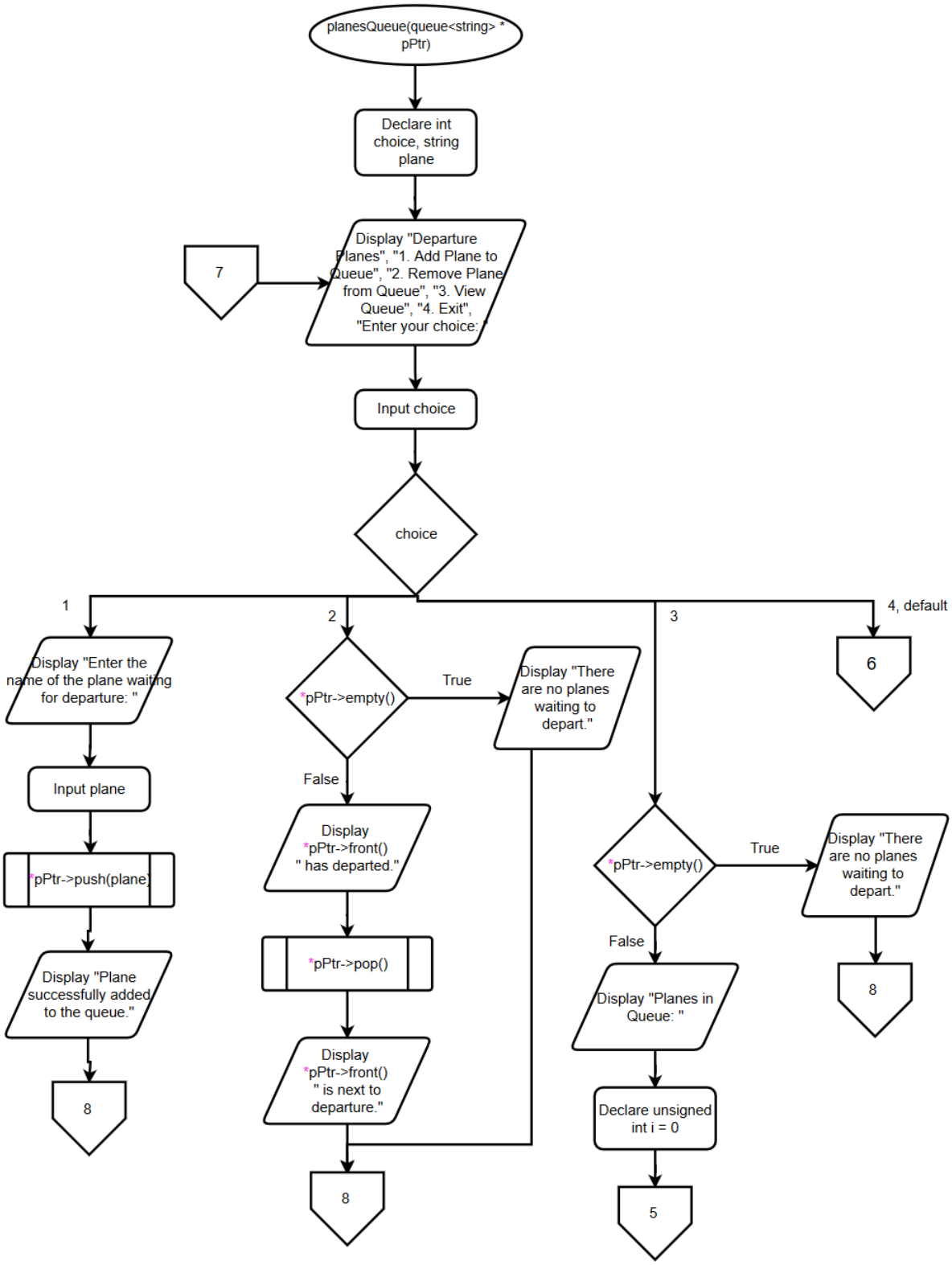
The flowchart for the Northwoods Airport, Inc.'s airport management system is as follows:

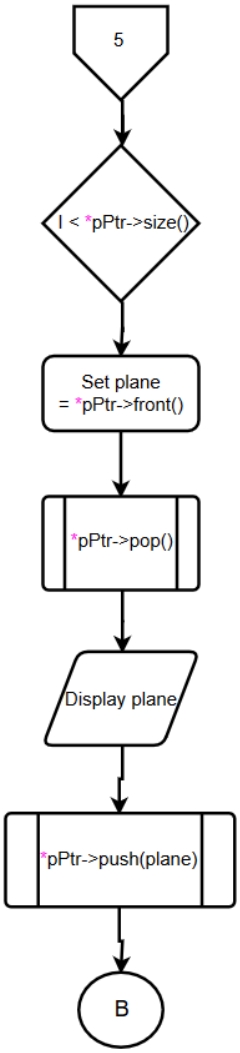


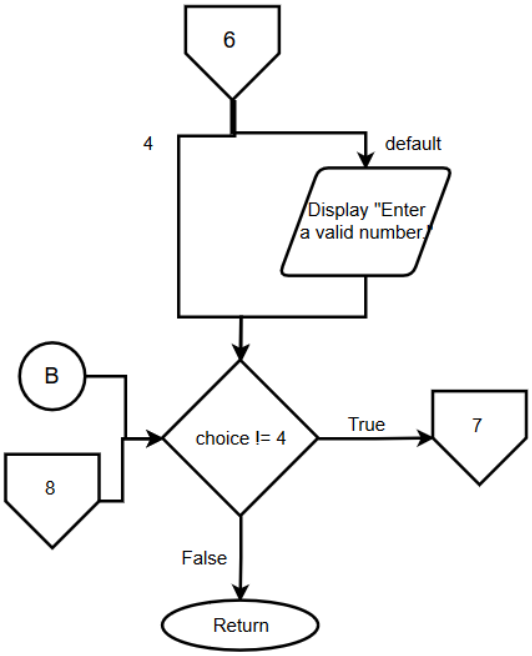

1
menuOption
7, 8, default
1
2
3
4
5
6
planesQueue(planesPtr)
viewEmployees(Employees)
binPacking()
planesLand(pLPtr)
securirytClear(securityPtr)
flightDistances()
2
3

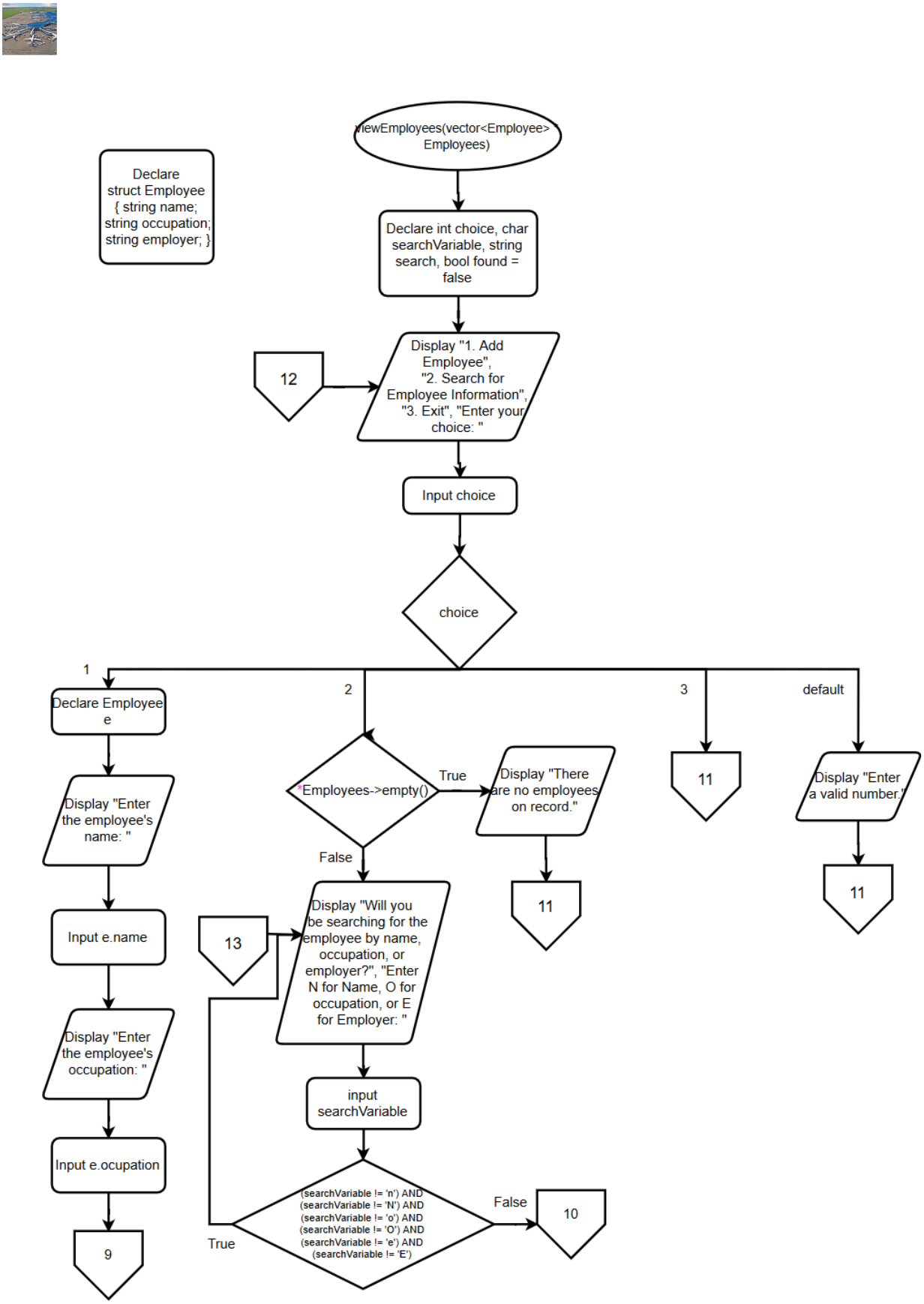

2
7
8
default
Display "Are you sure you want to exit (Y for Yes, N for No): "
Input answer
answer != 'y' AND answer != 'Y' AND answer != 'n' AND answer != 'N'
True
False
answer == 'y' OR answer == 'Y'
Set menuOption = 0
True
3
Display "Below are the numbers corresponding to the same ones in the main menu. After those", "numbers (below), is information about what the user can do if said number is chosen.", "1. The user can add or remove airplanes waiting to land or depart from the runway.", "2. View employee information stored (Name, Occupation, Employer).", "3. The user can add a customer's baggage in appropriate containers per flight.", "4. The user can how many planes are waiting to land.", "5. View the names of the customers waiting for security clearance.", "6. View the total distance a flight will from a specific destination hub to a specific arrival hub.", "7. The user will be asked if they wish to continue to exit the program. If the user enters 'y' or 'Y', then the user will be logged out. If the input is 'n' or 'N', the program will take the user to the main menu."
Display "Please enter the appropriate number (See the top of the menu)."

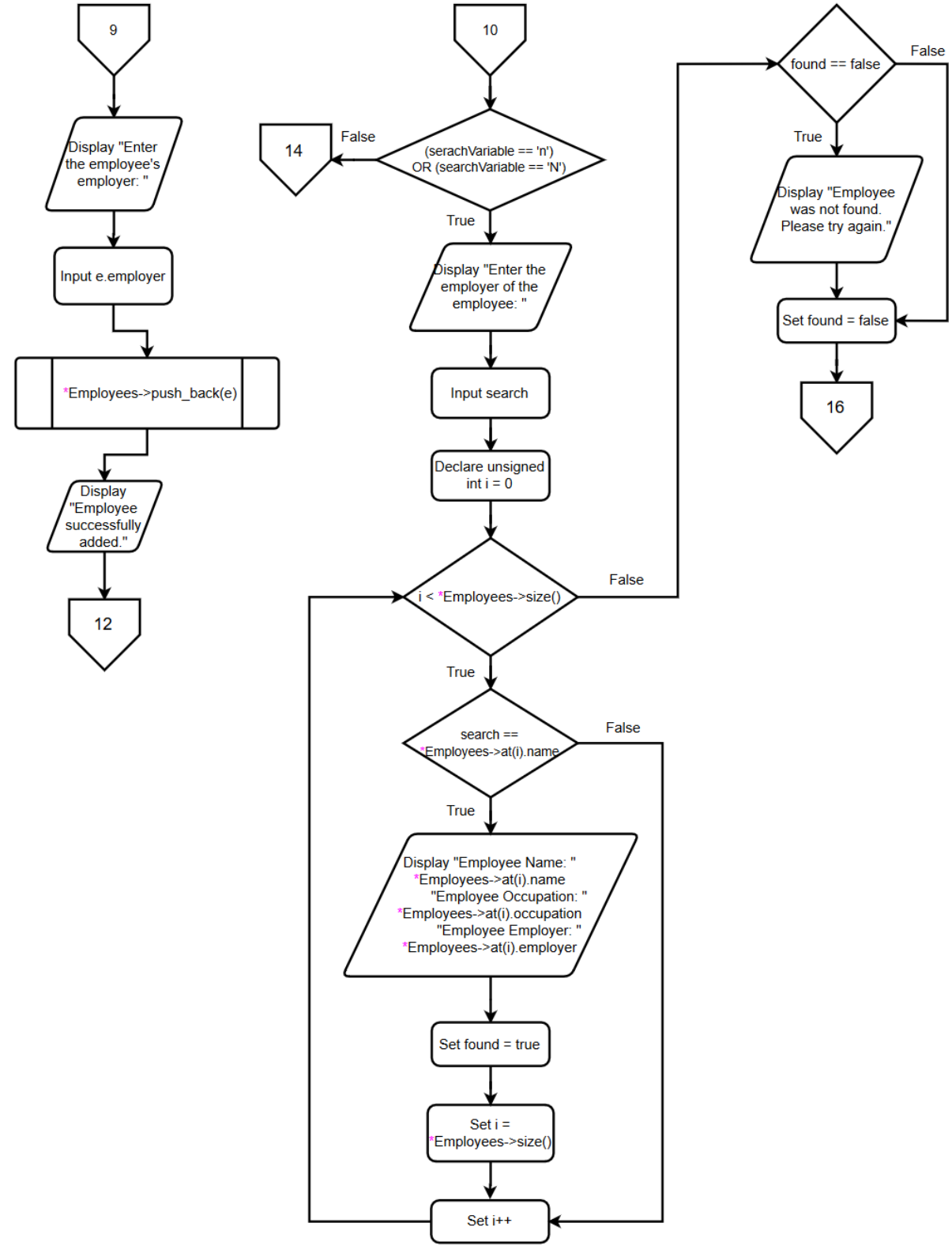

3
menuOption != 7
True
4
False
Display "Successfully Logged Out"
End

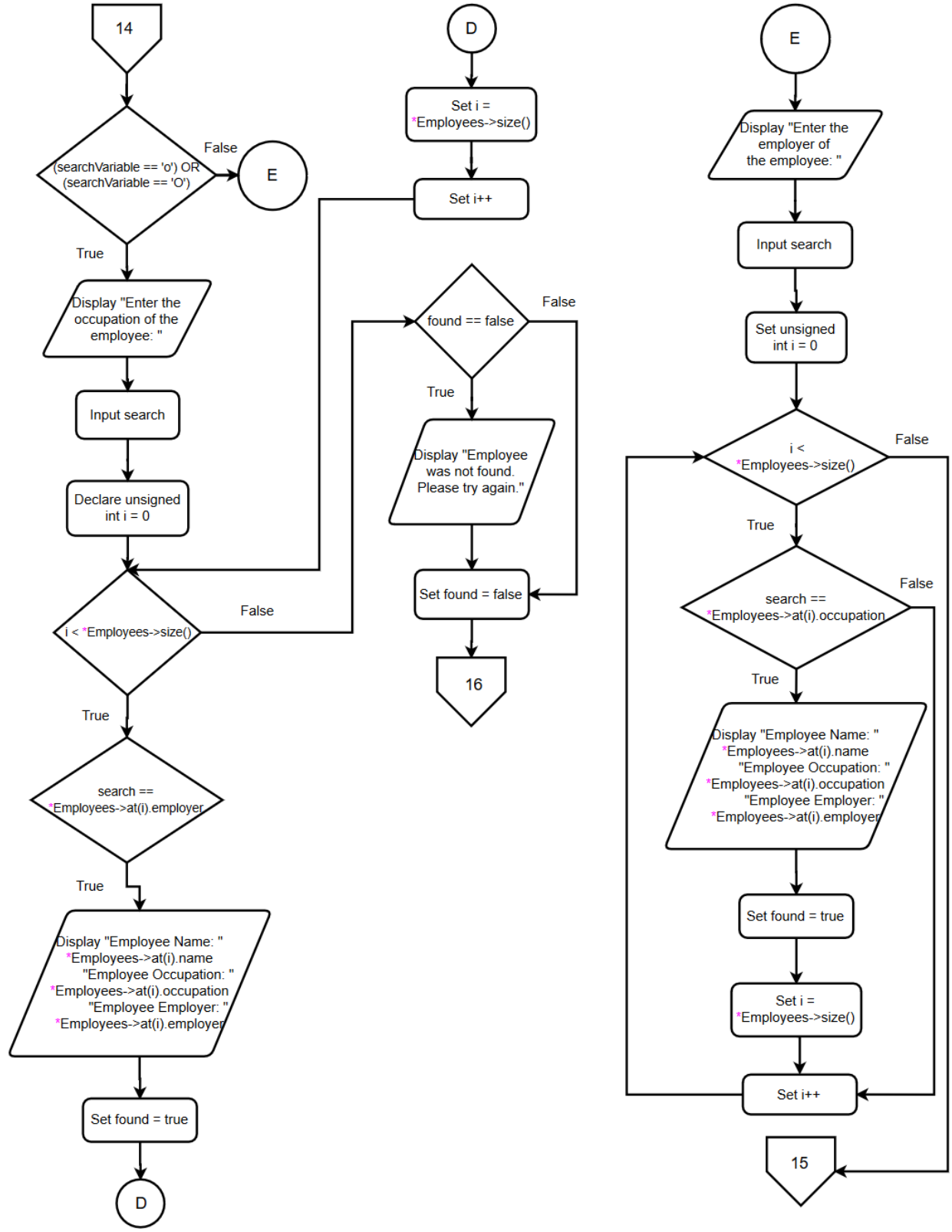

planesQueue(queue<string> * pPtr)
Declare int choice, string plane
7
Display "Departure Planes", "1. Add Plane to Queue", "2. Remove Plane from Queue", "3. View Queue", "4. Exit", "Enter your choice: "
Input choice
choice
1
Display "Enter the name of the plane waiting for departure: "
Input plane
*pPtr->push(plane)
Display "Plane successfully added to the queue."
8
2
*pPtr->empty()
True
Display "There are no planes waiting to depart."
False
Display *pPtr->front() " has departed."
*pPtr->pop()
Display *pPtr->front() " is next to departure."
8
3
*pPtr->empty()
True
Display "There are no planes waiting to depart."
8
False
Display "Planes in Queue: "
Declare unsigned int i = 0
5
4, default
6

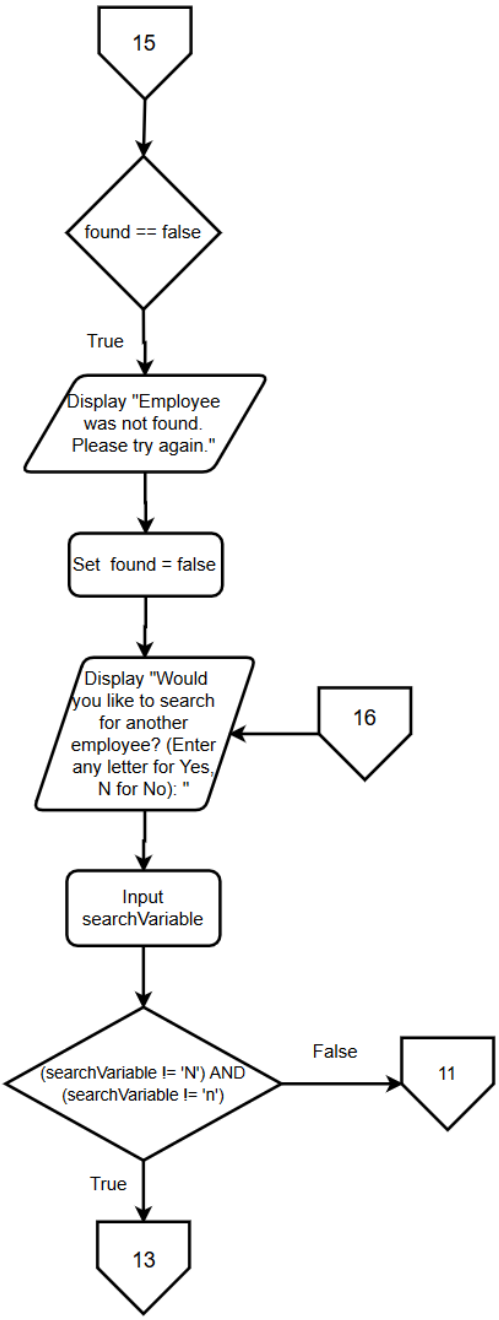

5
I < *pPtr->size()
Set plane
= *pPtr->front()
*pPtr->pop()
Display plane
*pPtr->push(plane)
B

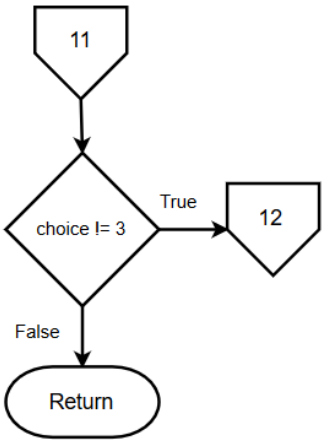

6
4
default
Display "Enter
a valid number"
B
choice I= 4
True
7
8
False
Return

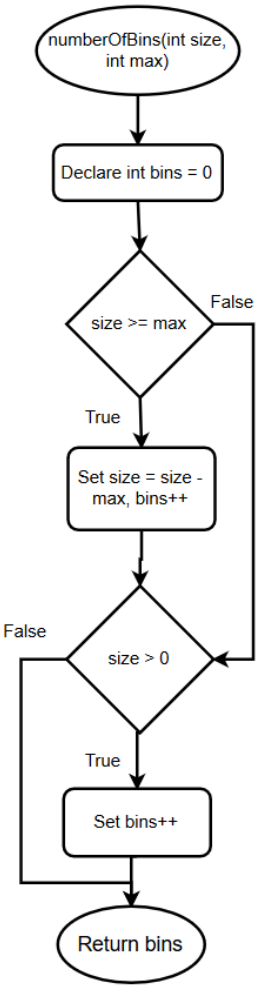

ViewEmployees(vector<Employee> Employees)
Declare struct Employee { string name; string occupation; string employer; }
Declare int choice, char searchVariable, string search, bool found = false
12
Display "1. Add Employee", "2. Search for Employee Information", "3. Exit", "Enter your choice: "
Input choice
choice
1
Declare Employee e
Display "Enter the employee's name: "
Input e.name
Display "Enter the employee's occupation: "
Input e.ocupation
9
2
*Employees->empty()
True
Display "There are no employees on record."
11
False
13
Display "Will you be searching for the employee by name, occupation, or employer?", "Enter N for Name, O for occupation, or E for Employer: "
input searchVariable
(searchVariable != 'n') AND (searchVariable != 'N') AND (searchVariable != 'o') AND (searchVariable != 'O') AND (searchVariable != 'e') AND (searchVariable != 'E')
True
False
10
3
11
default
Display "Enter a valid number."
11

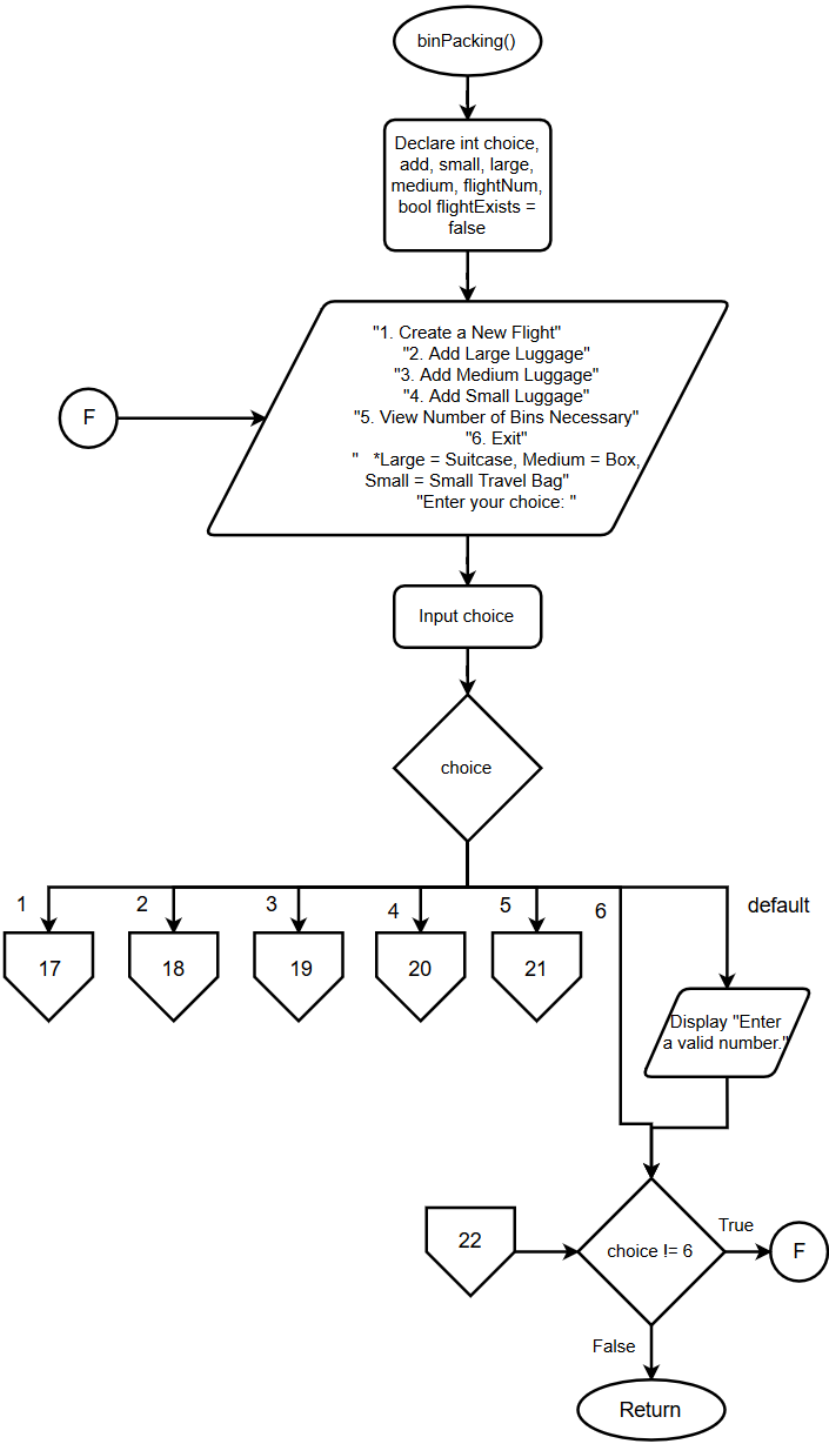

9
Display "Enter the employee's employer: "
Input e.employer
*Employees->push_back(e)
Display "Employee successfully added."
12
10
14
False
(serachVariable == 'n') OR (searchVariable == 'N')
True
Display "Enter the employer of the employee: "
Input search
Declare unsigned int i = 0
i < *Employees->size()
False
True
search == *Employees->at(i).name
False
True
Display "Employee Name: " *Employees->at(i).name "Employee Occupation: " *Employees->at(i).occupation "Employee Employer: " *Employees->at(i).employer
Set found = true
Set i = *Employees->size()
Set i++
found == false
False
True
Display "Employee was not found. Please try again."
Set found = false
16

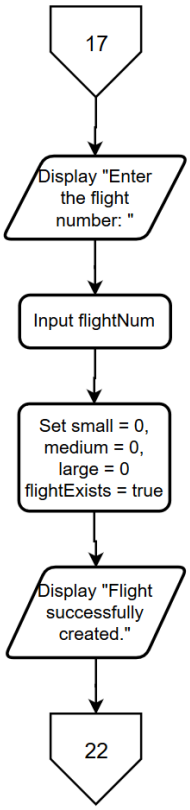

14
(searchVariable == 'o') OR (searchVariable == 'O')
False
E
True
Display "Enter the occupation of the employee: "
Input search
Declare unsigned int i = 0
i < *Employees->size()
False
True
search == *Employees->at(i).employer
True
Display "Employee Name: " *Employees->at(i).name "Employee Occupation: " *Employees->at(i).occupation "Employee Employer: " *Employees->at(i).employer
Set found = true
D
D
Set i = *Employees->size()
Set i++
found == false
False
True
Display "Employee was not found. Please try again."
Set found = false
16
E
Display "Enter the employer of the employee: "
Input search
Set unsigned int i = 0
i < *Employees->size()
False
True
search == *Employees->at(i).occupation
False
True
Display "Employee Name: " *Employees->at(i).name "Employee Occupation: " *Employees->at(i).occupation "Employee Employer: " *Employees->at(i).employer
Set found = true
Set i = *Employees->size()
Set i++
15

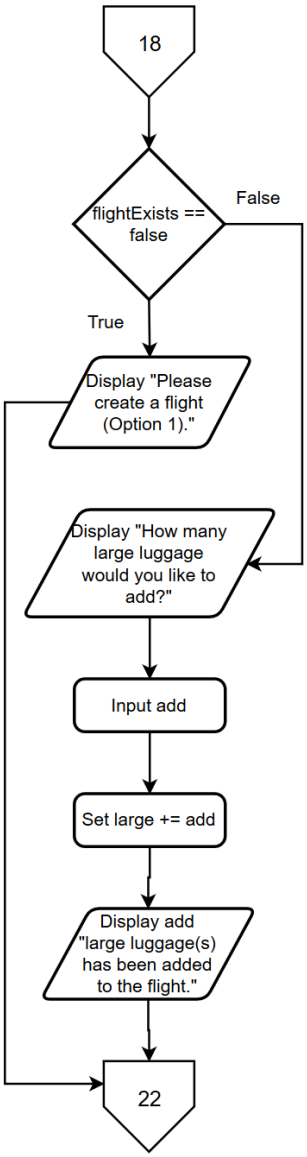

15
found == false
True
Display "Employee was not found. Please try again."
Set found = false
Display "Would you like to search for another employee? (Enter any letter for Yes, N for No): "
16
Input searchVariable
(searchVariable != 'N') AND (searchVariable != 'n')
False
11
True
13

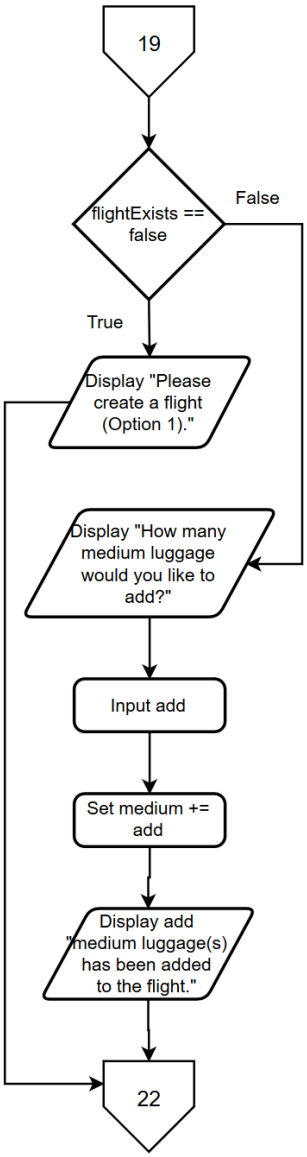

11
choice != 3
True
12
False
Return

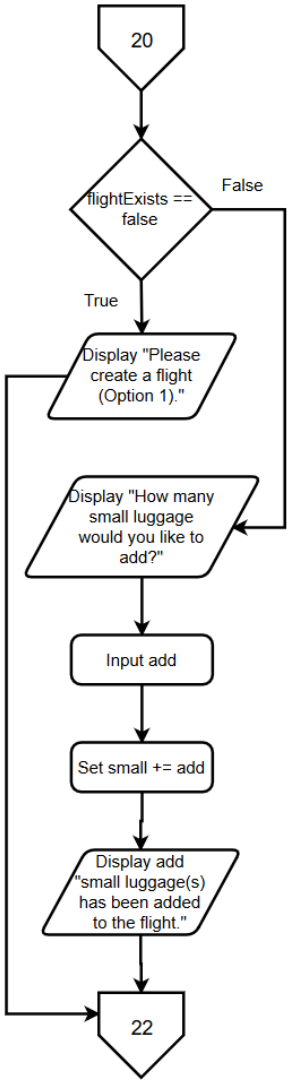

numberOfBins(int size, int max)
Declare int bins = 0
size >= max
False
True
Set size = size - max, bins++
False
size > 0
True
Set bins++
Return bins

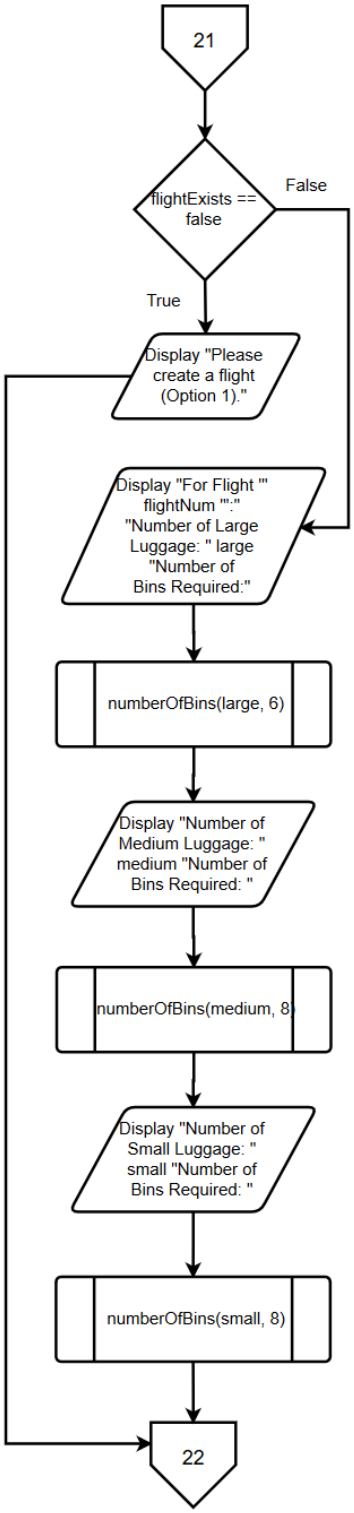

binPacking()
Declare int choice, add, small, large, medium, flightNum, bool flightExists = false
F
"1. Create a New Flight"
"2. Add Large Luggage"
"3. Add Medium Luggage"
"4. Add Small Luggage"
"5. View Number of Bins Necessary"
"6. Exit"
" *Large = Suitcase, Medium = Box, Small = Small Travel Bag"
"Enter your choice: "
Input choice
choice
1
2
3
4
5
6
default
17
18
19
20
21
Display "Enter a valid number."
22
choice != 6
True
F
False
Return

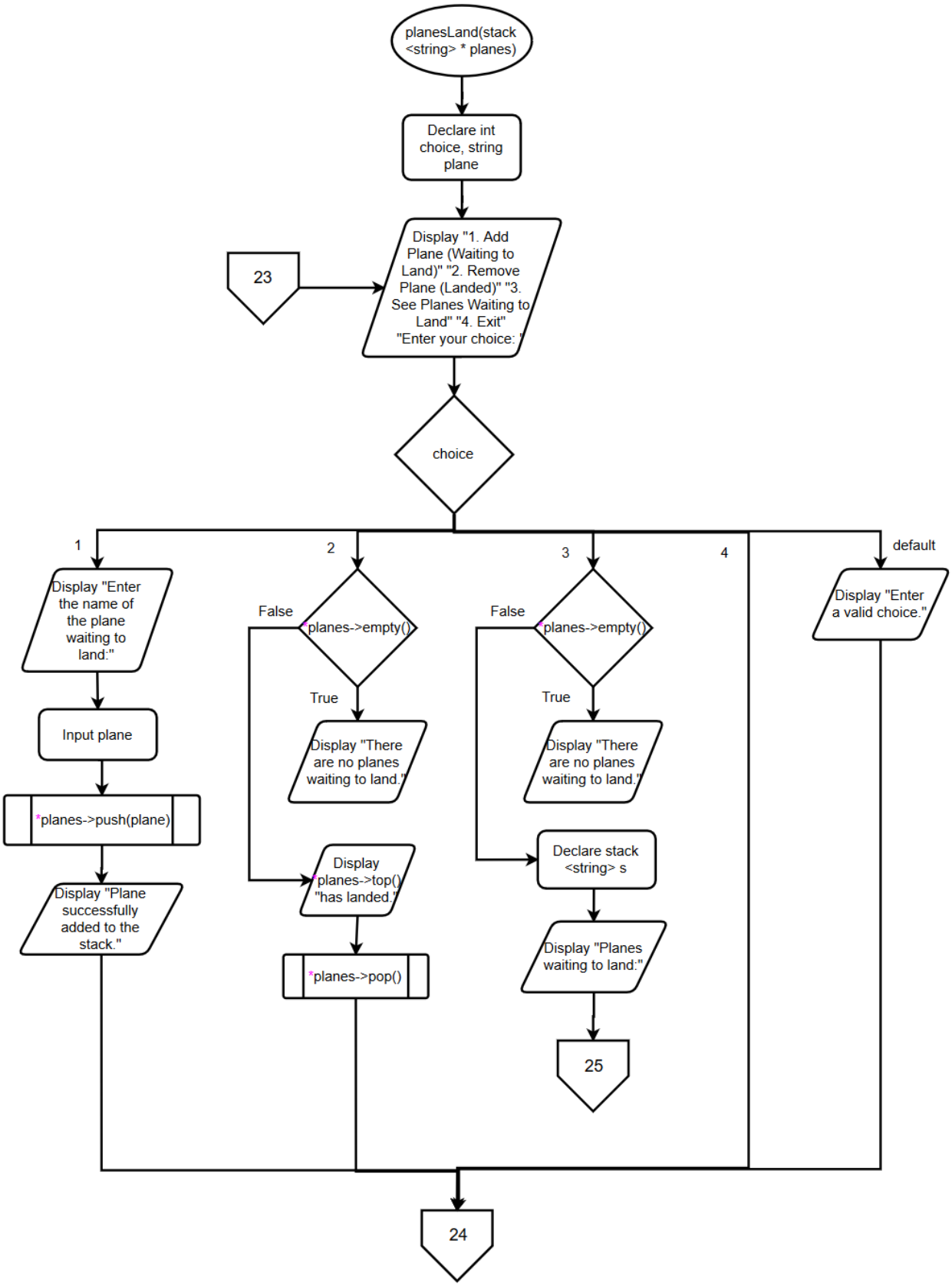

17
Display "Enter the flight number: "
Input flightNum
Set small = 0, medium = 0, large = 0 flightExists = true
Display "Flight successfully created."
22

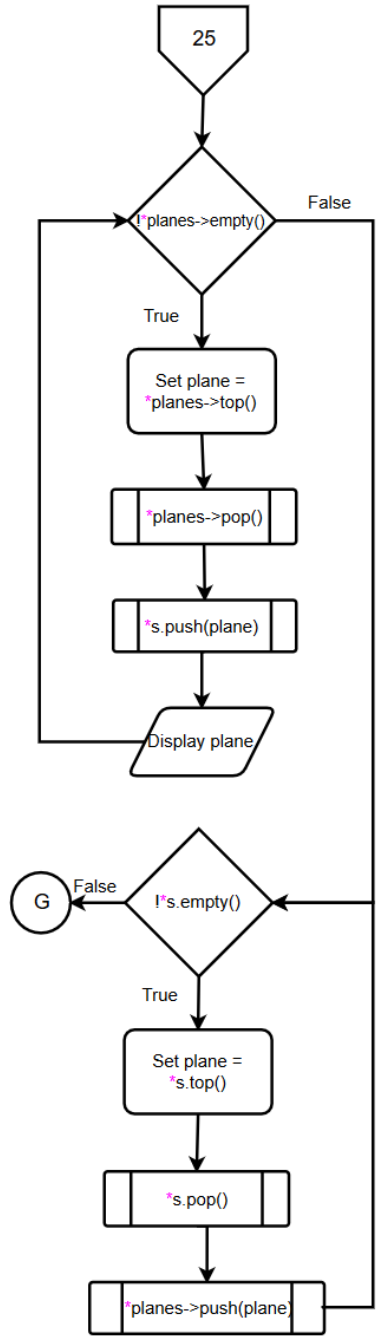

18
flightExists == false
False
True
Display "Please create a flight (Option 1)."
Display "How many large luggage would you like to add?"
Input add
Set large += add
Display add "large luggage(s) has been added to the flight."
22

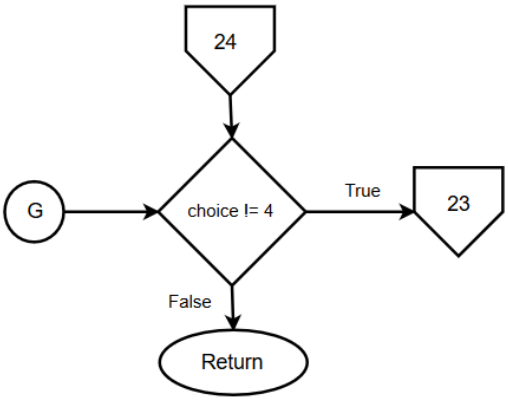

19
flightExists == false
False
True
Display "Please create a flight (Option 1)."
Display "How many medium luggage would you like to add?"
Input add
Set medium += add
Display add "medium luggage(s) has been added to the flight."
22

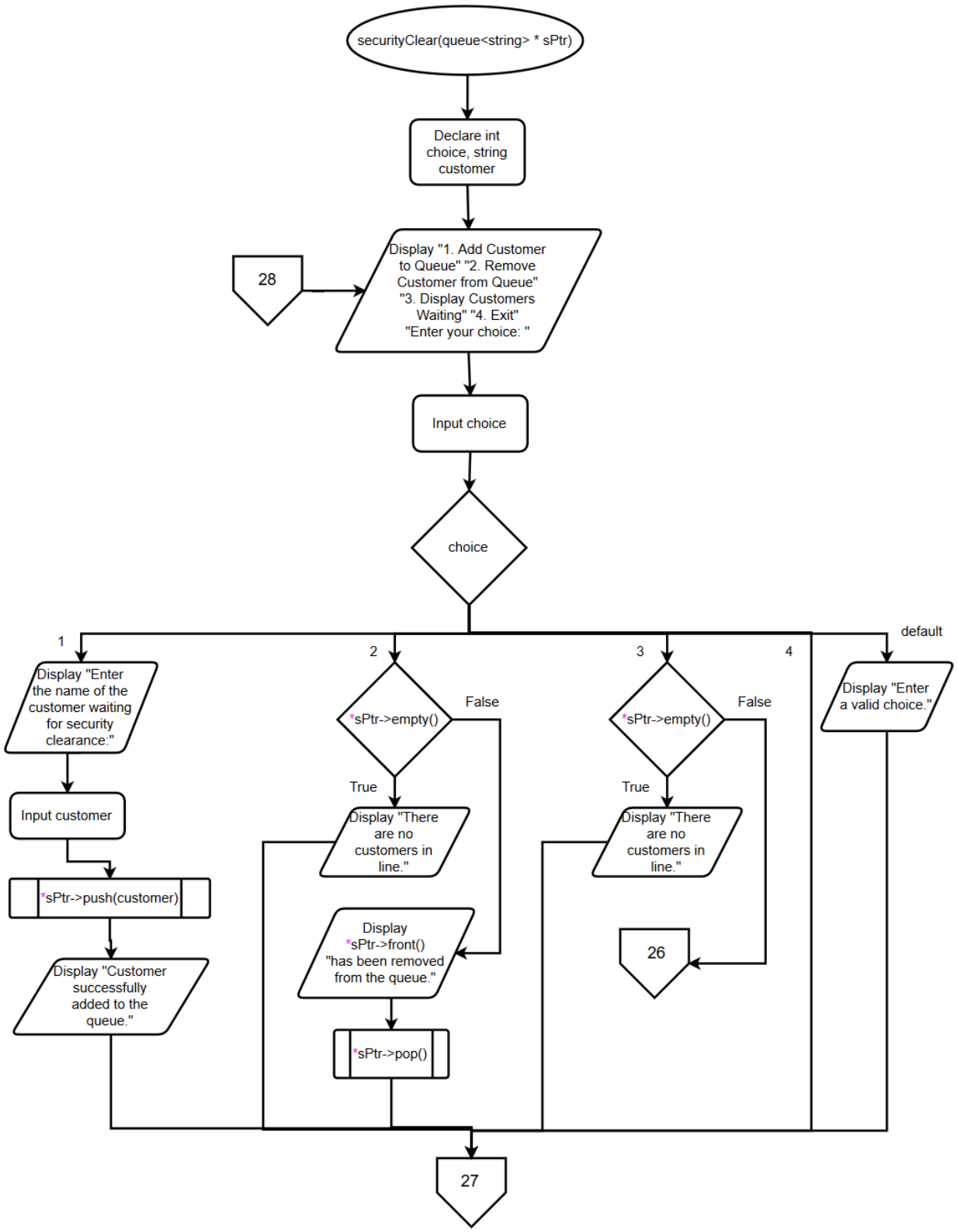

20
flightExists == false
False
True
Display "Please create a flight (Option 1)."
Display "How many small luggage would you like to add?"
Input add
Set small += add
Display add "small luggage(s) has been added to the flight."
22

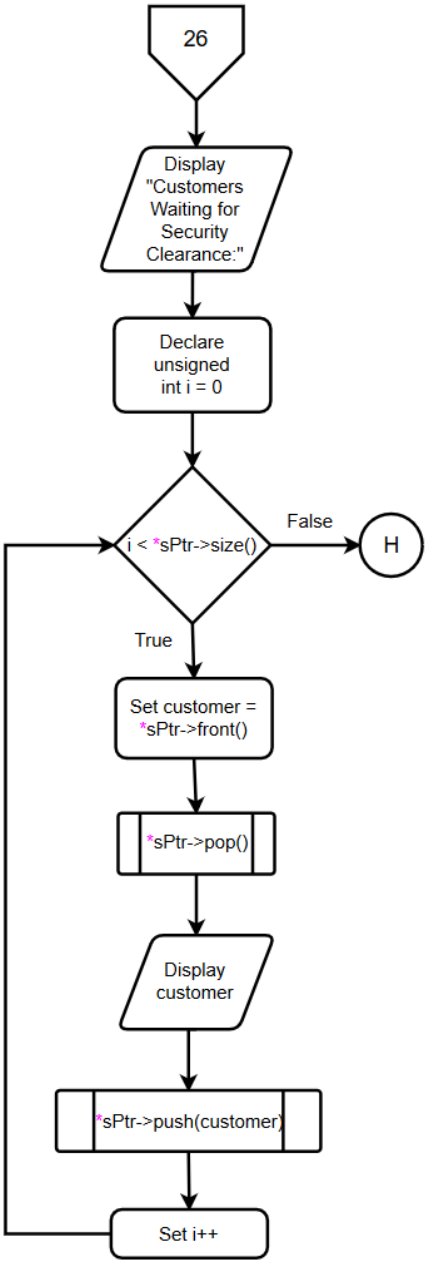

21
flightExists == false
False
True
Display "Please create a flight (Option 1)."
Display "For Flight '" flightNum "':" "Number of Large Luggage: " large "Number of Bins Required:"
numberOfBins(large, 6)
Display "Number of Medium Luggage: " medium "Number of Bins Required: "
numberOfBins(medium, 8)
Display "Number of Small Luggage: " small "Number of Bins Required: "
numberOfBins(small, 8)
22

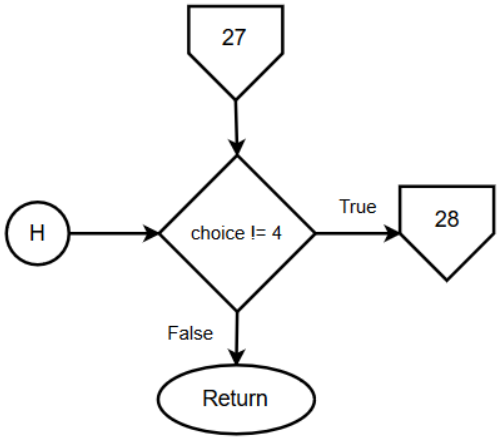

planesLand(stack <string> * planes)
Declare int choice, string plane
23
Display "1. Add Plane (Waiting to Land)" "2. Remove Plane (Landed)" "3. See Planes Waiting to Land" "4. Exit" "Enter your choice: "
choice
1
2
3
4
default
Display "Enter the name of the plane waiting to land:"
Input plane
*planes->push(plane)
Display "Plane successfully added to the stack."
False
*planes->empty()
True
Display "There are no planes waiting to land."
Display *planes->top() "has landed."
*planes->pop()
False
*planes->empty()
True
Display "There are no planes waiting to land."
Declare stack <string> s
Display "Planes waiting to land:"
25
Display "Enter a valid choice."
24


25
!*planes->empty()
False
True
Set plane =
*planes->top()
*planes->pop()
*s.push(plane)
Display plane
G
False
!*s.empty()
True
Set plane =
*s.top()
*s.pop()
*planes->push(plane)

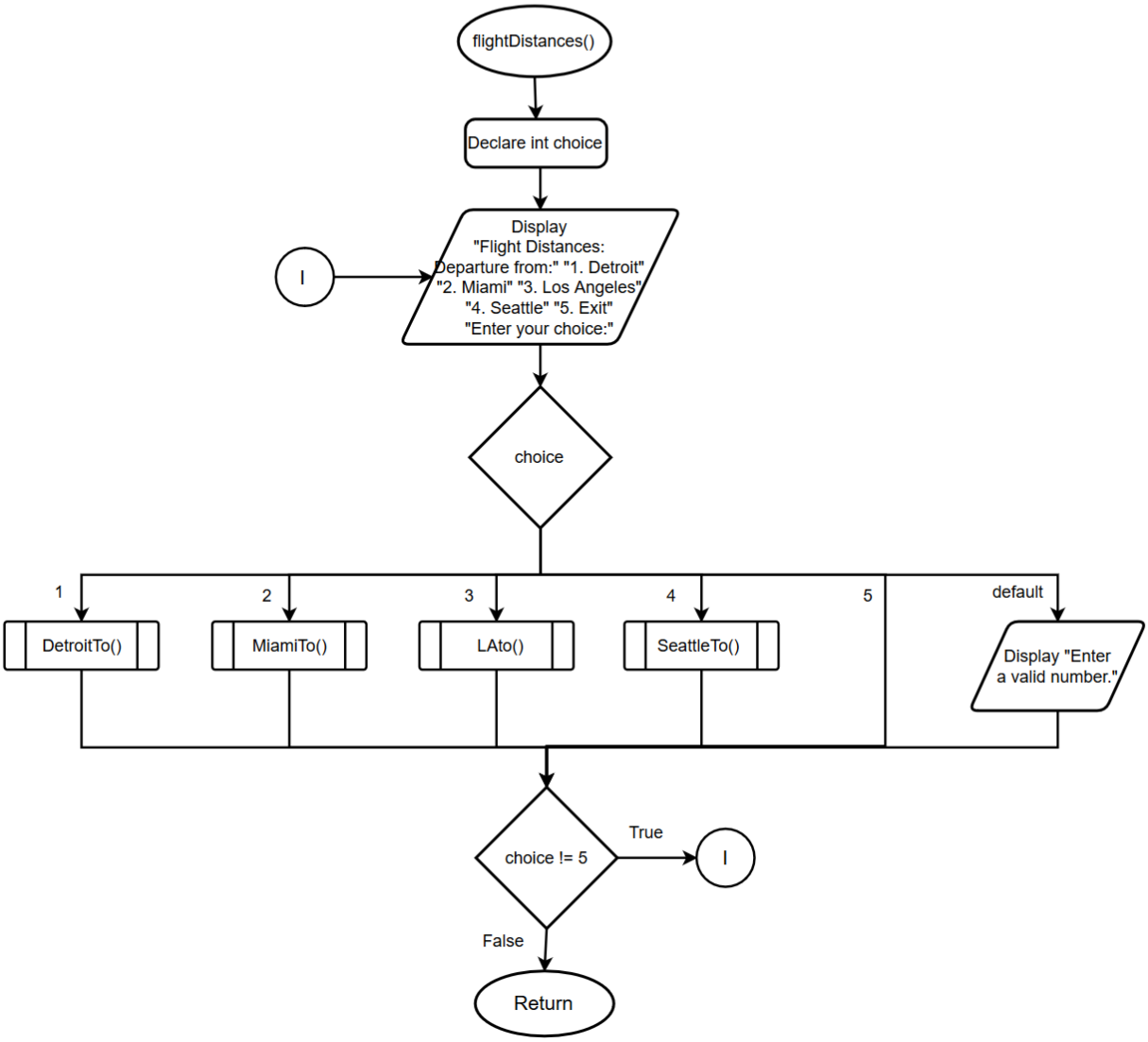

24
G
choice != 4
True
23
False
Return

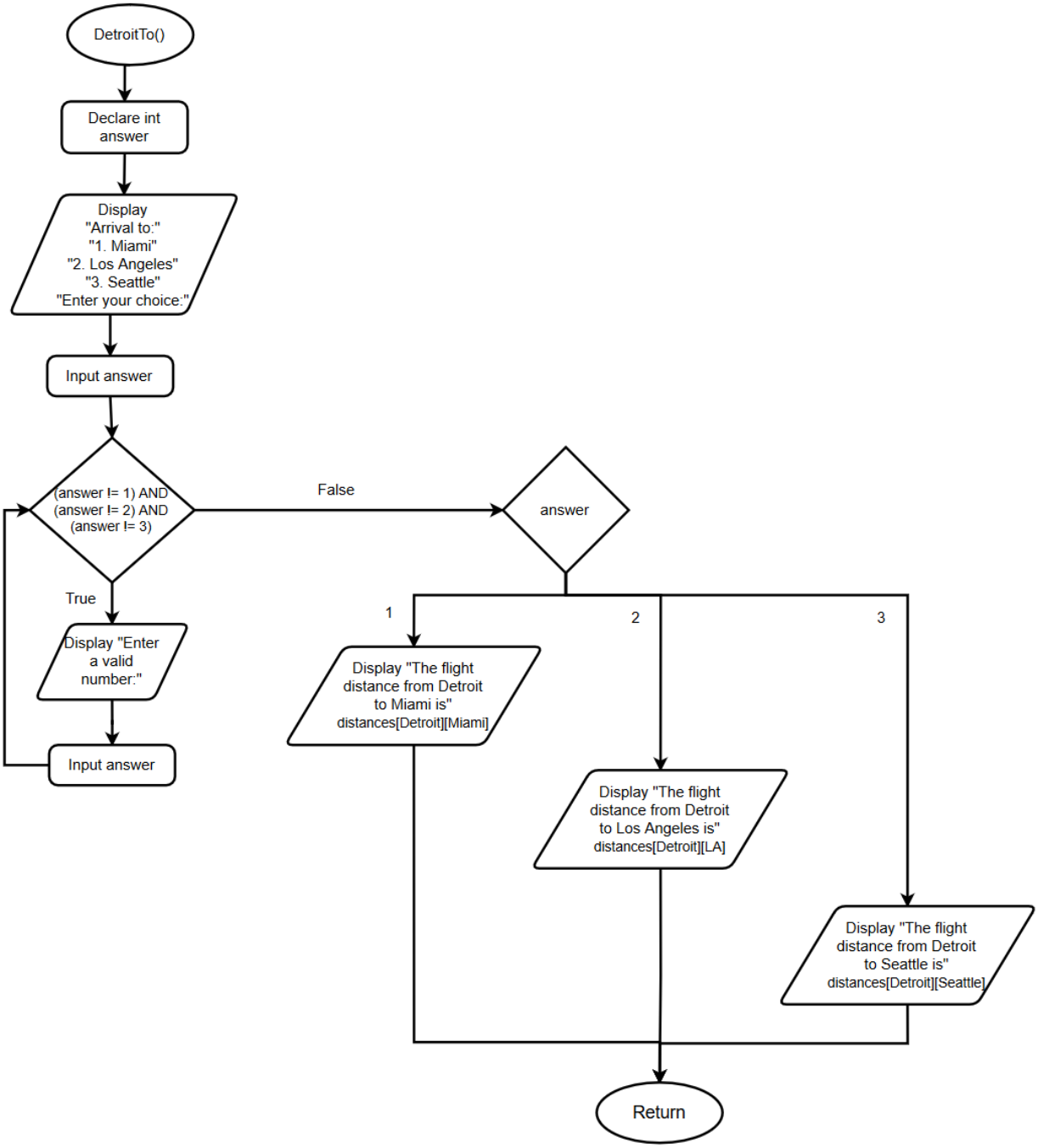

securityClear(queue<string> * sPtr)
Declare int choice, string customer
28
Display "1. Add Customer to Queue" "2. Remove Customer from Queue" "3. Display Customers Waiting" "4. Exit" "Enter your choice: "
Input choice
choice
1
2
3
4
default
Display "Enter the name of the customer waiting for security clearance:"
Input customer
*sPtr->push(customer)
Display "Customer successfully added to the queue."
*sPtr->empty()
False
True
Display "There are no customers in line."
Display *sPtr->front() "has been removed from the queue."
*sPtr->pop()
*sPtr->empty()
False
True
Display "There are no customers in line."
26
Display "Enter a valid choice."
27

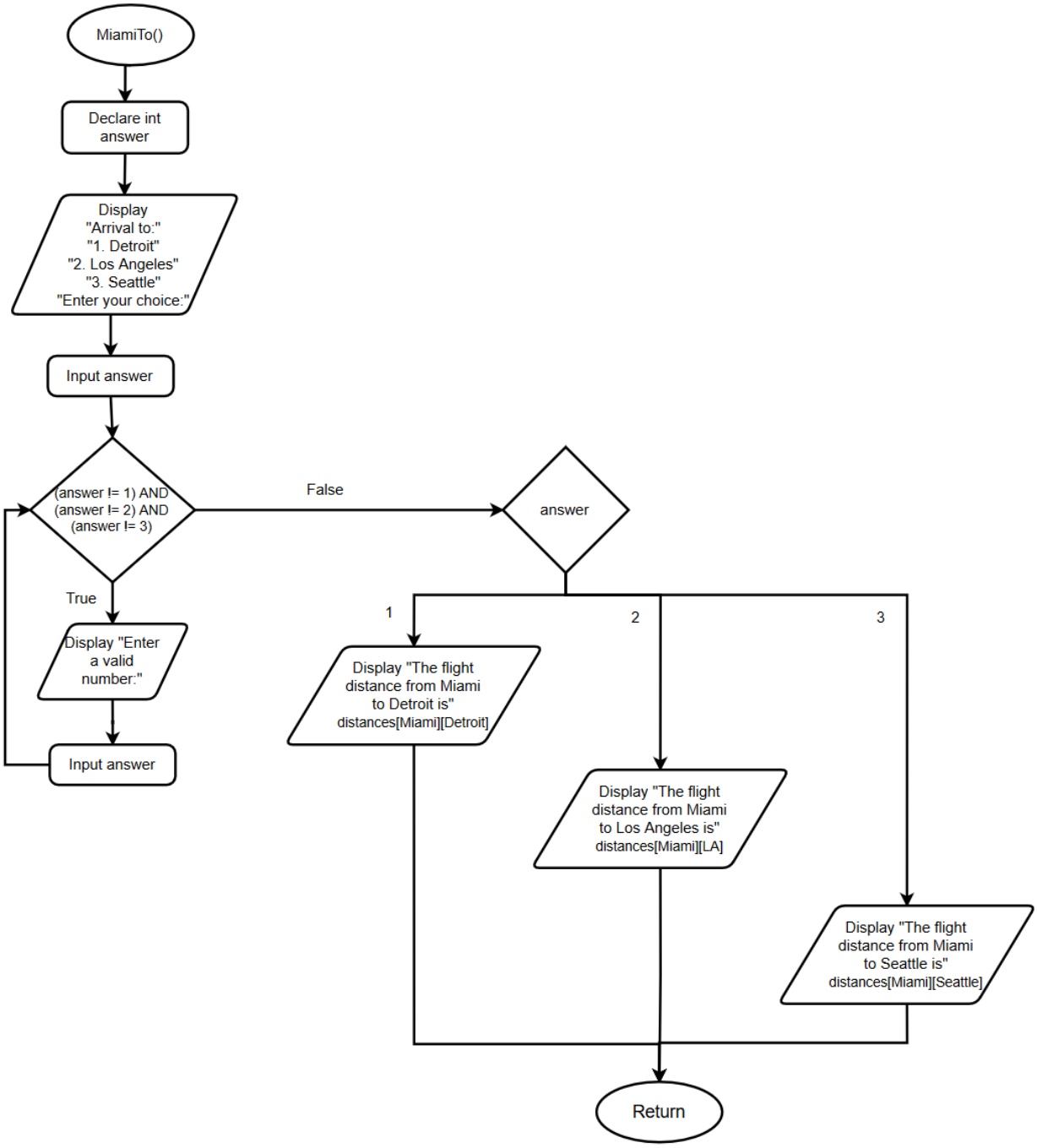

26
Display "Customers Waiting for Security Clearance:"
Declare unsigned int i = 0
i < *sPtr->size()
False
H
True
Set customer = *sPtr->front()
*sPtr->pop()
Display customer
*sPtr->push(customer)
Set i++

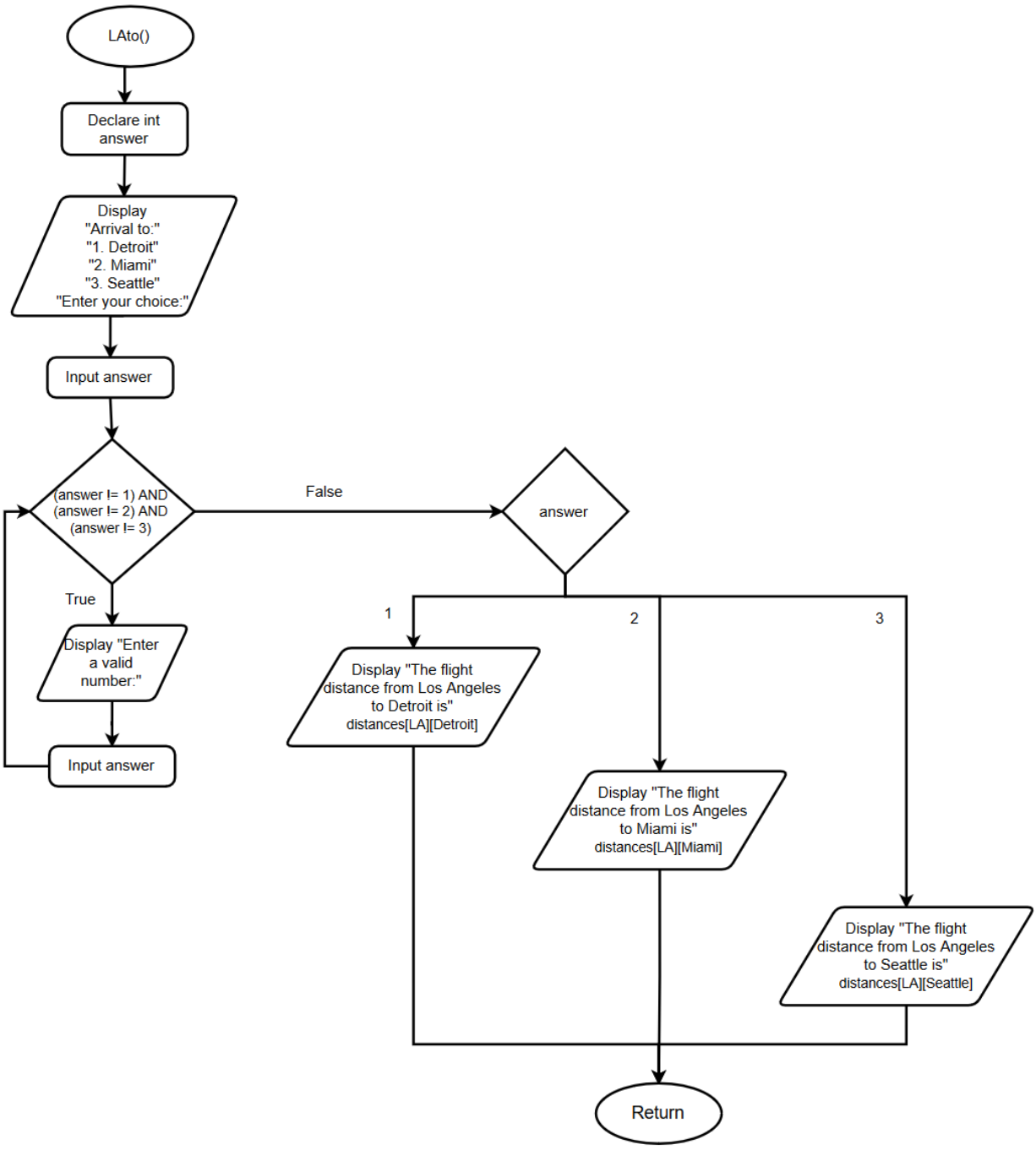

27
H
choice != 4
True
28
False
Return

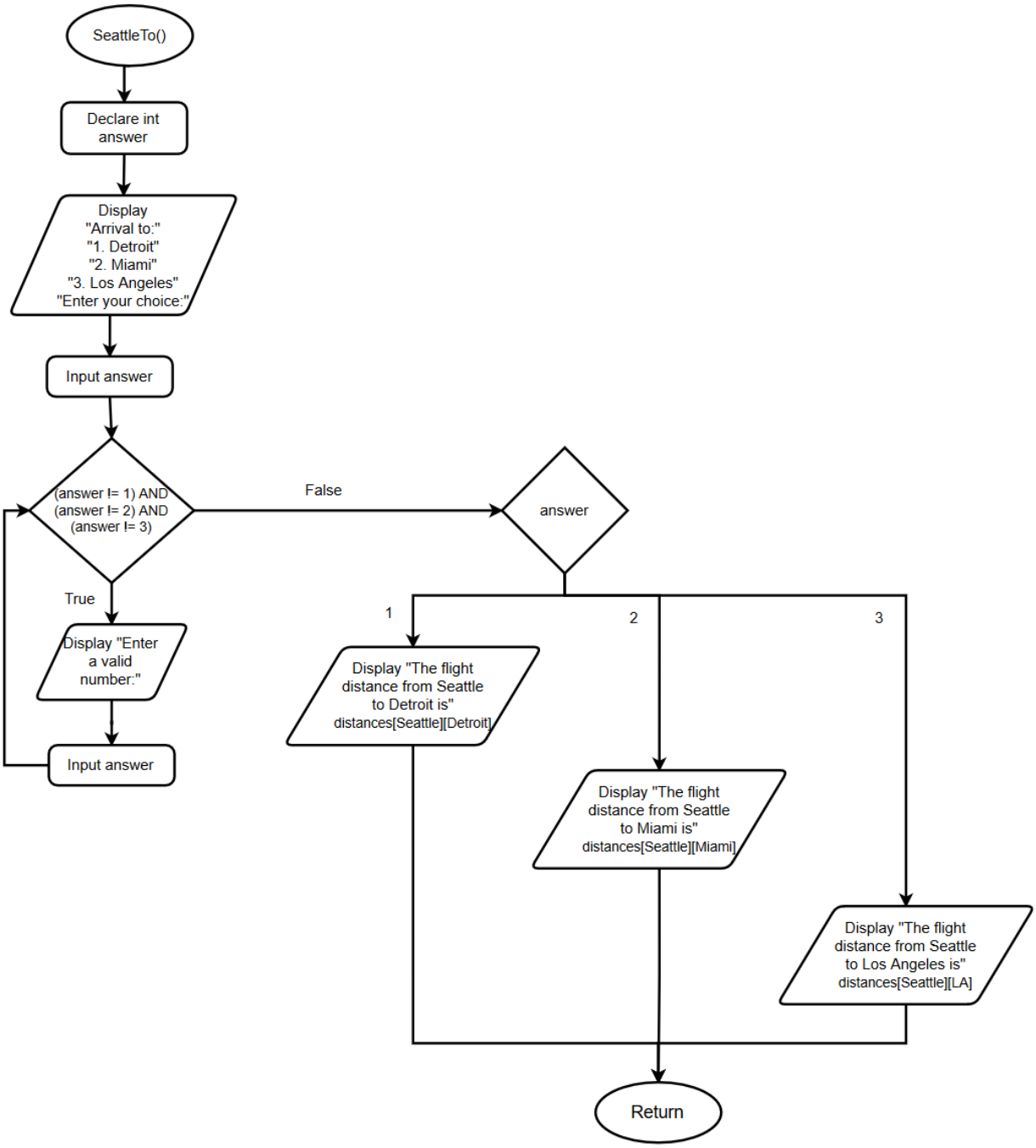
Declare enum Flights {Detroit, Miami, La, Seattle}, const int SIZE = 4, static string distances[SIZE][SIZE] =
{ " ", "1153 mi/1855 km", "1983 mi/3192 km", "1938 mi/3119 km" },
{ "1153 mi/1855 km", " ", "2339 mi/3764 km", "2734 mi/4399 km" },
{ "1983 mi/3192 km", "2339 mi/3764 km", " ", "959 mi/1544 km" },
{ "1938 mi/3119 km", "2734 mi/4399 km", "959 mi/1544 km", " " }




DetroitTo()
Declare int answer
Display "Arrival to:" "1. Miami" "2. Los Angeles" "3. Seattle" "Enter your choice:"
Input answer
(answer != 1) AND (answer != 2) AND (answer != 3)
False
answer
True
Display "Enter a valid number:"
Input answer
1
2
3
Display "The flight distance from Detroit to Miami is" distances[Detroit][Miami]
Display "The flight distance from Detroit to Los Angeles is" distances[Detroit][LA]
Display "The flight distance from Detroit to Seattle is" distances[Detroit][Seattle]
Return


MiamiTo()
Declare int answer
Display "Arrival to:" "1. Detroit" "2. Los Angeles" "3. Seattle" "Enter your choice:"
Input answer
(answer != 1) AND (answer != 2) AND (answer != 3)
False
True
Display "Enter a valid number:"
Input answer
answer
1
2
3
Display "The flight distance from Miami to Detroit is" distances[Miami][Detroit]
Display "The flight distance from Miami to Los Angeles is" distances[Miami][LA]
Display "The flight distance from Miami to Seattle is" distances[Miami][Seattle]
Return


LAto()
Declare int answer
Display "Arrival to:" "1. Detroit" "2. Miami" "3. Seattle" "Enter your choice:"
Input answer
(answer != 1) AND (answer != 2) AND (answer != 3)
False
True
Display "Enter a valid number:"
Input answer
answer
1
2
3
Display "The flight distance from Los Angeles to Detroit is" distances[LA][Detroit]
Display "The flight distance from Los Angeles to Miami is" distances[LA][Miami]
Display "The flight distance from Los Angeles to Seattle is" distances[LA][Seattle]
Return


SeattleTo()
Declare int answer
Display "Arrival to:" "1. Detroit" "2. Miami" "3. Los Angeles" "Enter your choice:"
Input answer
(answer != 1) AND (answer != 2) AND (answer != 3)
False
answer
True
Display "Enter a valid number:"
Input answer
1
2
3
Display "The flight distance from Seattle to Detroit is" distances[Seattle][Detroit]
Display "The flight distance from Seattle to Miami is" distances[Seattle][Miami]
Display "The flight distance from Seattle to Los Angeles is" distances[Seattle][LA]
Return

UML Representation: Use Case

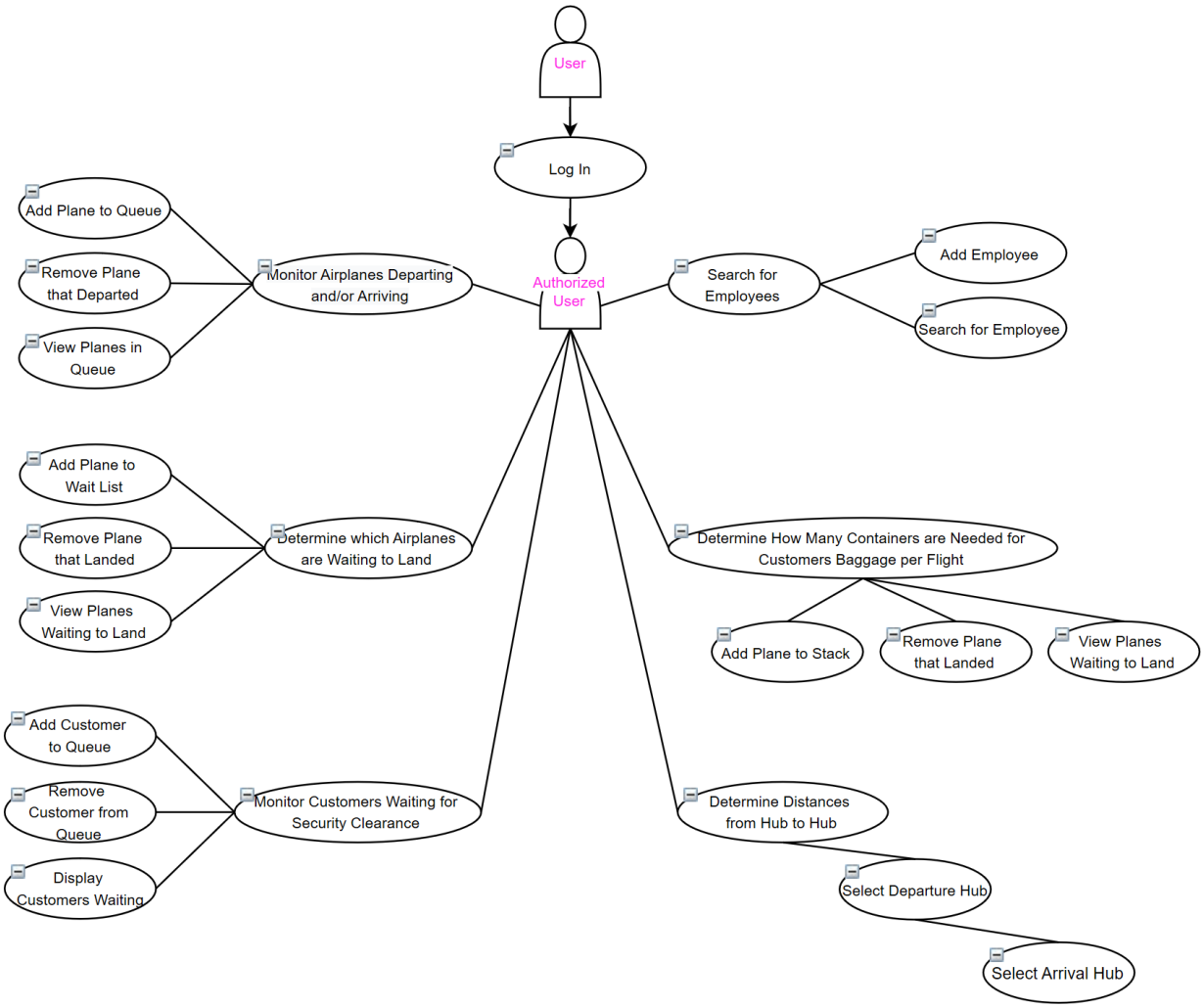


Chart for Designated Data Structures and their Functions

Data Structure	Function	Employee Tasks
Array	**Where needed for data structure creation and use**	
Bin	**Construct an array to hold the elements (i.e., the individual luggage); use another array to designate the bins**	**Determine which bins have reached capacity and therefore are ready to load on the plane**
Stack	**Construct stack of airplanes waiting to land**	**Display planes in stack; remove the planes from the stack once they have landed**
Queue	**Construct queue for planes waiting to take off or land**	**Allow authorized employee to add planes to the queue for taking off and then to remove the departing plane from the queue**
Connected Graph	**Construct a connected graph to show connectivity between hubs**	**Determine distance between destination and arrival hubs**
Struct	**Construct a struct for Employee data; construct a struct object to access the employee**	**Insert data on Employee; Search for Employee, and display data on Employee**

Data Spreadsheet – for The Northwoods Airport Management System

Menu Item for	Employee List	Airline plane Destination from Chicago [check distances]	Customer Baggage	Airplanes waiting to land	Airplanes waiting to take off	Passengers waiting for security clearance
Name or Data	1. John Smith 2. Bob Sands 3. Amelia Hill 4. Mike Apreza	1. Detroit 2. Miami 3. Los Angeles 4. Seattle	1. For flight 675: 10 suitcases, 7 boxes, 20 small travel bags Container capacity size: Large 6 Medium 8 Small 5	1. Pennsylvania Air Flight 987 2. Atlantic Flight 345 3. TransOceanic Flight 753 4. Jaguar Airlines Flight 221	1. Atlantic Flight 675. 2. Union Flyers 403 3. Michigan Flight Transport	1. Sally Diaz 2. Mike Manos 3. Ethan Storm
Occupation	1. Pilot 2. Mechanic 3. Airport Security Mgr. 4. IT					
Employer	1. Johnson Airlines 2. Johnson Airlines 3. Northwoods 4. COD					

Complete C++ Code (with comment statements), including Header, Implementation, and Driver file(s).

Driver File

```
/*
Name:  Mike Apreza
File:  menu.cpp
Date:  12/12/2019
       Updated: 02/15/2020
Class: CIS 2542-002
*/

// STL and Files
#include <iostream>
#include <string>
#include <vector>
#include <queue>
#include <stack>
#include "planesTakeOffAndLand.cpp"
#include "viewEmployees.cpp"
#include "baggage.cpp"
#include "planesLanding.cpp"
#include "securityClearance.cpp"
#include "flightTotalDistance.cpp"

// The main function
int main()
{
	// Declare string variable to hold password
	std::string password;
	// Declare integer variable to hold choice option for switch statement
	int menuOption = 0;
	// Declare character variable to hold choice whether or not user wants to quit using
the system
	char answer = 'N';
	// Declare vector of Employee structs and pointer variable to point at a vector of
structs
	std::vector<Employee> employeesVector;
	std::vector<Employee> * Employees = &employeesVector;
	// Declare queue of strings for customers waiting for security clearance and pointer to
queue
	std::queue<std::string> security;
	std::queue<std::string> * securityPtr = &security;
	// Declare queue of strings for airplanes waiting to take off and pointer to queue
	std::queue<std::string> planes;
	std::queue<std::string> * planesPtr = &planes;
	// Declare stack of strings for airplanes waiting to land and pointer to stack
	std::stack <std::string> planesLanding;
	std::stack <std::string> * pLPtr = &planesLanding;

	// Display first menu and ask user for password to access menu for authorized users
	do
	{
		system("CLS");
		std::cout << "\t\tNorthwoods Airport, Inc. Management System\n" << "\t\t\tPlease
enter the password.\n"
			<< "\t  Having trouble? Contact the IT department at 630-942-2800.\n\n"
			<< "\t\tPassword: ";
		std::cin >> password;
	} while (password != "delta");
```

```
	/*
                                                                    OVERALL SUMMARY OF
MENU
		This menu displays a welcome message and informs the user of how s/he can
execute certain functions.
		Below that is a series of lines displaying a number followed by an action the
user can execute.
		At the very bottom is a place for the user to enter the number corresponding to
the action that will
		execute is said number is chosen. There are 8 options (numbers 1 through 8). If
any other number is
		entered, the program will inform the user of their mistake and the menu will
appear again as if the
		user has just logged in. Should the user want to exit, they will be asked for
confirmation.
	*/
	do
	{
		system("CLS");
		std::cout << "\t\t\tWelcome.\n\tPlease select a menu option to get started.\n\t
(Enter the corresponding number.)\n\n";
		std::cout << "1. Add or Remove Airplane(s) (Arrival and Departure)\n"
			<< "2. View Employee Information\n"
			<< "3. Add Customer Baggage\n"
			<< "4. Add or Remove Planes that are Waiting to Land\n"
			<< "5. Add or Remove Customer to Wait for Security Clearance\n"
			<< "6. View Total Distance a Plane Travels from One Place to Another\n"
			<< "7. Exit\n"
			<< "8. Information about the Options\n";
		std::cout << " Option: ";
		std::cin >> menuOption;

		switch (menuOption)
		{
			case 1:
			{
				planesQueue(planesPtr);
				break;
			}
			case 2:
			{
				viewEmployees(Employees);
				break;
			}
			case 3:
			{
				binPacking();
				break;
			}
			case 4:
			{
				planesLand(pLPtr);
				break;
			}
			case 5:
			{
				securityClear(securityPtr);
				break;
			}
			case 6:
```

```
                        {
                                flightDistances();
                                break;
                        }
                        case 7:
                        {
                                // Ask user if they are sure they want to exit
                                do
                                {
                                        std::cout << " Are you sure you want to exit? (Y for Yes,
N for No): ";
                                        std::cin >> answer;
                                } while ((answer != 'y') && (answer != 'Y') && (answer != 'n') &&
(answer != 'N'));

                                // Exit system if user selects Yes, otherwise, go back to main
menu
                                if ((answer == 'y') || (answer == 'Y'))
                                        break;
                                else
                                {
                                        menuOption = 0;
                                        break;
                                }
                        }
                        case 8:
                        {
                                system("CLS");
                                std::cout << "Below are the numbers corresponding to the same
ones in the main menu. After those\n"
                                        << "numbers (below), is information about what the user
can do if said number is chosen.\n\n";
                                std::cout << "1. The user can add or remove airplanes waiting to
land or depart from the runway.\n"
                                        << "2. View employee information stored (Name, Occupation,
Employer).\n"
                                        << "3. The user can add a customer's baggage in
appropriate containers per flight.\n"
                                        << "4. The user can how many planes are waiting to
land.\n"
                                        << "5. View the names of the customers waiting for
security clearance.\n"
                                        << "6. View the total distance a flight will from a
specific destination hub to a specific arrival hub.\n"
                                        << "7. The user will be asked if they wish to continue to
exit the program. If the user enters 'y' or 'Y',\n"
                                        << "   then the user will be logged out. If the input is
'n' or 'N', the program will take the user to the main menu.\n\n";
                                system("PAUSE");
                                break;
                        }
                        default:
                        {
                                // informs user of their mistake
                                system("CLS");
                                std::cout << "Please enter the appropriate number (See the top of
the menu).\n";
                                system("PAUSE");
                                break;
                        }
                }
```

```
        } while (menuOption != 7);

        // Inform use that logging out was successful
        system("CLS");
        std::cout << "\n\t\tSuccessfully Logged Out\n";

        return 0;
}
```

Implementation Files

planesTakeOffAndLand.cpp

```
/*
Name:  Mike Apreza
File:  planesTakeOffAndLand.cpp
Date:  12/12/2019
Class: CIS 2542-002
*/

#include <iostream>
#include <string>
#include <queue>

/*
	This function takes a pointer to a queue of strings and lets the authorized employee
	to add a plane to the queue of planes waiting to departure, remove the next plane from
	the queue, and view all the planes in the queue.
*/
static void planesQueue(std::queue<std::string> * pPtr)
{
	// Declare integer variable to hold choice
	int choice;
	// Declare string variale to hold name of airplane
	std::string plane;

	// Loop to iterate for as long as user wishes to take action on the planes
	do
	{
		system("CLS");
		std::cout << "\tDeparture Planes\n"
			<< "1. Add Plane to Queue\n"
			<< "2. Remove Plane from Queue\n"
			<< "3. View Queue\n"
			<< "4. Exit\n"
			<< "Enter your choice: ";
		std::cin >> choice;

		// Switch statement
		switch (choice)
		{
			case 1:
			{
				std::cin.ignore();
				std::cout << "Enter the name of the plane waiting for departure:
";
				std::getline(std::cin, plane);
				pPtr->push(plane);
				std::cout << "Plane successfully added to the queue.\n";
				system("PAUSE");
				break;
			}
			case 2:
			{
				if (pPtr->empty())
				{
					std::cout << "There are no planes waiting to depart.\n";
				}
				else
```

```
                        {
                                std::cout << pPtr->front() << " has departed.\n";
                                pPtr->pop();
                                std::cout << pPtr->front() << " is next to departure.\n";
                        }
                        system("PAUSE");
                        break;
                }
                case 3:
                {
                        if (pPtr->empty())
                        {
                                std::cout << "There are no planes waiting in queue.\n";
                        }
                        else
                        {
                                std::cout << "\nPlanes in Queue:\n";
                                for (unsigned int i = 0; i < pPtr->size(); i++)
                                {
                                        plane = pPtr->front();
                                        pPtr->pop();
                                        std::cout << plane << std::endl;
                                        pPtr->push(plane);
                                }
                        }
                        system("PAUSE");
                        break;
                }
                case 4:
                        break;
                default:
                {
                        std::cout << "Enter a valid number.\n";
                        system("PAUSE");
                        break;
                }
        }
    } while (choice != 4);
}
```

viewEmployees.cpp

```
/*
Name:  Mike Apreza
File:  viewEmployees.cpp
Date:  12/12/2019
Class: CIS 2542-002
*/

#include <iostream>
#include <string>
#include <vector>

// Struct to hold information about employee (name, occupation, employer)
struct Employee
{
	std::string name;
	std::string occupation;
	std::string employer;
};

// This function takes a pointer to a vector of structs
static void viewEmployees(std::vector<Employee> * Employees)
{
	// Declare int variable to hold choice for switch statements
	int choice;
	// Declare character variable for choice 3 in switch statement (search by)
	char searchVariable;
	// Declare string variable for name, occupation, or employer
	std::string search;
	// Declare Boolean variable
	bool found = false;

	do
	{
		system("CLS");
		// Display Menu
		std::cout << "1. Add Employee\n"
			<< "2. Search for Employee Information\n"
			<< "3. Exit\n"
			<< "Enter your choice: ";
		std::cin >> choice;

		// Switch statement
		switch (choice)
		{
			case 1:
			{
				system("CLS");
				// Create Employee and ask for the employee's information
				Employee e;
				std::cout << "Enter the employee's name: ";
				std::cin.ignore();
				std::getline(std::cin, e.name);
				std::cout << "Enter the employee's occupation: ";
				std::getline(std::cin, e.occupation);
				std::cout << "Enter the employee's employer: ";
				std::getline(std::cin, e.employer);
				// Add employee to vector
				Employees->push_back(e);
				std::cout << "Employee successfully added.\n";
```

```
                        system("PAUSE");
                        break;
                }
                case 2:
                {
                        if (Employees->empty())
                        {
                                std::cout << "There are no employees on record.\n";
                                system("PAUSE");
                                break;
                        }
                        else
                        {
                                // This loop will iterate as long as the user wishes to
search for an employee
                                do
                                {
                                        system("CLS");
                                        std::cout << "Will you be searching for the
employee by name, occupation, or employer?\n";
                                        do
                                        {
                                                std::cout << "Enter N for Name, O for
occupation, or E for Employer: ";
                                                std::cin >> searchVariable;
                                        } while ((searchVariable != 'n') &&
(searchVariable != 'N') && (searchVariable != 'o') && (searchVariable != 'O') &&
(searchVariable != 'e') && (searchVariable != 'E'));

                                        // Let user enter what they they wish to search
for according to searchVariable
                                        if ((searchVariable == 'n') || (searchVariable ==
'N'))
                                        {
                                                std::cout << "Enter the name of the
employee: ";
                                                std::cin.ignore();
                                                std::getline(std::cin, search);
                                                for (unsigned int i = 0; i < Employees-
>size(); i++)
                                                {
                                                        if (search == Employees->at(i).name)
                                                        {
                                                                std::cout << "Employee Name:
" << Employees->at(i).name << std::endl;
                                                                std::cout << "Employee
Occupation: " << Employees->at(i).occupation << std::endl;
                                                                std::cout << "Employee
Employer: " << Employees->at(i).employer << std::endl;
                                                                found = true;
                                                                system("PAUSE");
                                                                i = Employees->size();
                                                        }
                                                }
                                                if (found == false)
                                                        std::cout << "Employee was not
found. Please try again.\n";
                                                found = false;
                                        }
                                        else if ((searchVariable == 'o') ||
(searchVariable == 'O'))
```

```
                                        {
                                                std::cout << "Enter the occupation of the
employee: ";
                                                std::cin.ignore();
                                                std::getline(std::cin, search);
                                                for (unsigned int i = 0; i < Employees-
>size(); i++)
                                                {
                                                        if (search == Employees-
>at(i).occupation)
                                                        {
                                                                std::cout << "Employee Name:
" << Employees->at(i).name << std::endl;
                                                                std::cout << "Employee
Occupation: " << Employees->at(i).occupation << std::endl;
                                                                std::cout << "Employee
Employer: " << Employees->at(i).employer << std::endl;
                                                                found = true;
                                                                system("PAUSE");
                                                                i = Employees->size();
                                                        }
                                                }
                                                if (found == false)
                                                        std::cout << "Employee was not
found. Please try again.\n";
                                                found = false;

                                        }
                                        else
                                        {
                                                std::cout << "Enter the employer of the
employee: ";
                                                std::cin.ignore();
                                                std::getline(std::cin, search);
                                                for (unsigned int i = 0; i < Employees-
>size(); i++)
                                                {
                                                        if (search == Employees-
>at(i).employer)
                                                        {
                                                                std::cout << "Employee Name:
" << Employees->at(i).name << std::endl;
                                                                std::cout << "Employee
Occupation: " << Employees->at(i).occupation << std::endl;
                                                                std::cout << "Employee
Employer: " << Employees->at(i).employer << std::endl;
                                                                found = true;
                                                                system("PAUSE");
                                                                i = Employees->size();
                                                        }
                                                }
                                                if (found == false)
                                                        std::cout << "Employee was not
found. Please try again.\n";
                                                found = false;
                                        }

                                        // Ask user if they wish to search for another
employee
                                        std::cout << "Would you like to search for another
employee? (Enter any letter for Yes, N for No): ";
```

```
                                        std::cin >> searchVariable;
                                    } while ((searchVariable != 'N') && (searchVariable != 
'n'));
                                    break;
                                }
                        }
                        case 3:
                                break;
                        default:
                        {
                                std::cout << "Enter a valid number.\n";
                                system("PAUSE");
                                break;
                        }
                }
        } while (choice != 3);
}
```

baggage.cpp

```
/*
Name:  Mike Apreza
File:  baggage.cpp
Date:  01/06/2020
Class: CIS 2542-002
*/

#include <iostream>

/* This function takes an integer values (number of luggage of a particular size
 and max value for container) and calculates the number of bins required.
   The number of bins is returned*/

static int numberOfBins(int size, int max)
{
	int bins = 0;
	while (size >= max)
	{
		size -= max;
		bins++;
	}
	if (size > 0)
		bins++;
	return bins;
}

/* This function allows the user to enter the amount of luggage there is for
	a specific flight depending on the size. As each element is added, it will
	also be put in it's appropraite bin. */
static void binPacking()
{
	// Create integer variable to hold choice for switch statement, number of baggage,
number of elements,
	// holder for 'small,' 'medium,' and 'large,' and flight number
	int choice, add, small, medium, large, flightNum;
	// Create boolean variable to hold if a flight has not been created yet
	bool flightExists = false;

	do
	{
		system("CLS");
		std::cout << "1. Create a New Flight\n"
			<< "2. Add Large Luggage\n"
			<< "3. Add Medium Luggage\n"
			<< "4. Add Small Luggage\n"
			<< "5. View Number of Bins Necessary\n"
			<< "6. Exit\n"
			<< "   *Large = Suitcase, Medium = Box, Small = Small Travel Bag\n"
			<< "Enter your choice: ";
		std::cin >> choice;

		// Switch statement
		switch (choice)
		{
			case 1:
			{
				std::cout << "Enter the flight number: ";
				std::cin >> flightNum;
				small = 0;
```

```
                        medium = 0;
                        large = 0;
                        std::cout << "Flight successfully created.\n";
                        flightExists = true;
                        system("PAUSE");
                        break;
                }
                case 2:
                {
                        if (flightExists == false)
                                std::cout << "Please create a flight (Option 1).\n";
                        else
                        {
                                std::cout << "How many large luggage would you like to
add? ";
                                std::cin >> add;
                                large += add;
                                std::cout << add << " large luggage(s) has been added to
the flight.\n";
                        }
                        system("PAUSE");
                        break;
                }
                case 3:
                {
                        if (flightExists == false)
                                std::cout << "Please create a flight (Option 1).\n";
                        else
                        {
                                std::cout << "How many medium luggage would you like to
add? ";
                                std::cin >> add;
                                medium += add;
                                std::cout << add << " medium luggage(s) has been added to
the flight.\n";
                        }
                        system("PAUSE");
                        break;
                }
                case 4:
                {
                        if (flightExists == false)
                                std::cout << "Please create a flight (Option 1).\n";
                        else
                        {
                                std::cout << "How many small luggage would you like to
add? ";
                                std::cin >> add;
                                small += add;
                                std::cout << add << " small luggage(s) has been added to
the flight.\n";
                        }
                        system("PAUSE");
                        break;
                }
                case 5:
                {
                        if (flightExists == false)
                                std::cout << "Please create a flight (Option 1).\n";
                        else
                        {
```

```
                                        std::cout << "\n\tFor Flight '" << flightNum << "':\n";
                                        // Call function to calculate number of bins and display
information
                                        std::cout << "Number of Large Luggage: " << large <<
"\tNumber of Bins Required: " << numberOfBins(large, 6) << std::endl;
                                        std::cout << "Number of Medium Luggage: " << medium <<
"\tNumber of Bins Required: " << numberOfBins(medium, 8) << std::endl;
                                        std::cout << "Number of Small Luggage: " << small <<
"\tNumber of Bins Required: " << numberOfBins(small, 5) << std::endl << std::endl;
                                }
                                system("PAUSE");
                                break;
                        }
                        case 6:
                                break;
                        default:
                        {
                                std::cout << "Enter a valid number.\n";
                                system("PAUSE");
                                break;
                        }
                }

        } while (choice != 6);
}
```

planesLanding.cpp

```
/*
Name:  Mike Apreza
File:  planesLanding.cpp
Date:  01/05/2020
Class: CIS 2542-002
*/

#include <iostream>
#include <stack>
#include <string>

// This function takes a pointer to a stack of strings
static void planesLand(std::stack <std::string> * planes)
{
	// Declare integer to hold choice
	int choice;
	// Declare string to hold name of plane
	std::string plane;

	do
	{
		system("CLS");
		// Display menu
		std::cout << "1. Add Plane (Waiting to Land)\n"
			<< "2. Remove Plane (Landed)\n"
			<< "3. See Planes Waiting to Land\n"
			<< "4. Exit\n"
			<< "Enter your choice: ";
		std::cin >> choice;

		// Switch statement
		// If option 2 or 3 is chosen, make sure it planes stack is not empty
		switch (choice)
		{
			case 1:
			{
				std::cin.ignore();
				std::cout << "Enter the name of the plane waiting to land: ";
				std::getline(std::cin, plane);
				planes->push(plane);
				std::cout << "Plane successfully added to the stack.\n";
				system("PAUSE");
				break;
			}
			case 2:
			{
				if (planes->empty())
				{
					std::cout << "There are no planes waiting to land.\n";
				}
				else
				{
					std::cout << planes->top() << " has landed.\n";
					planes->pop();
				}
				system("PAUSE");
				break;
			}
			case 3:
```

```
                        {
                                if (planes->empty())
                                {
                                        std::cout << "There are no planes waiting to land.\n";
                                }
                                else
                                {
                                        // Create stack of strings to hold planes (strings) from
planes stack
                                        std::stack <std::string> s;
                                        // Display stack
                                        std::cout << "\nPlanes waiting to land:\n";
                                        while (!planes->empty())
                                        {
                                                plane = planes->top();
                                                planes->pop();
                                                s.push(plane);
                                                std::cout << plane << std::endl;
                                        }
                                        // Fix planes stack
                                        while (!s.empty())
                                        {
                                                plane = s.top();
                                                s.pop();
                                                planes->push(plane);
                                        }
                                }
                                system("PAUSE");
                                break;
                        }
                        case 4:
                                break;
                        default:
                        {
                                std::cout << "Enter a valid choice.\n";
                                system("PAUSE");
                                break;
                        }
                }
        } while (choice != 4);
}
```

securityClearance.cpp

```
/*
Name:  Mike Apreza
File:  securityClearance.cpp
Date:  12/12/2019
Class: CIS 2542-002
*/

#include <iostream>
#include <string>
#include <queue>

static void securityClear(std::queue<std::string> * sPtr)
{
	// Declare integer variable to hold choice for switch statement
	int choice;
	// Declare string variable to hold name of a customer
	std::string customer;

	do
	{
		// Display menu
		system("CLS");
		std::cout << "1. Add Customer to Queue\n"
			<< "2. Remove Customer from Queue\n"
			<< "3. Display Customers Waiting\n"
			<< "4. Exit\n";
		std::cout << "Enter your choice: ";
		std::cin >> choice;

		switch (choice)
		{
			case 1:
			{
				std::cout << "Enter the name of the customer waiting for security
clearance: ";
				std::cin.ignore();
				std::getline(std::cin, customer);
				sPtr->push(customer);
				std::cout << "Customer successfully added to the queue.\n";
				system("PAUSE");
				break;
			}
			case 2:
			{
				if (sPtr->empty())
				{
					std::cout << "There are no customers in line.\n";
				}
				else
				{
					std::cout << sPtr->front() << " has been removed from the
queue.\n";
					sPtr->pop();
				}
				system("PAUSE");
				break;
			}
			case 3:
			{
```

```
                        if (sPtr->empty())
                        {
                                std::cout << "There are no customers in line.\n";
                        }
                        else
                        {
                                std::cout << "Customers Waiting for Security
Clearance:\n";
                                for (unsigned int i = 0; i < sPtr->size(); i++)
                                {
                                        // Remove customer from queue and display
                                        customer = sPtr->front();
                                        sPtr->pop();
                                        std::cout << customer << std::endl;
                                        // Add the same customer back into the queue
                                        sPtr->push(customer);
                                }
                        }
                        system("PAUSE");
                        break;
                }
                case 4:
                        break;
                default:
                {
                        std::cout << "Enter a valid number.\n";
                        system("PAUSE");
                        break;
                }
        }

        } while (choice != 4);
}
```

flightTotalDistance.cpp

```
/*
Name:  Mike Apreza
File:  flightTotalDistance.cpp
Date:  12/12/2019
Class: CIS 2542-002
*/

#include <iostream>
#include <string>

// Make enum global
enum Flights { Detroit, Miami, LA, Seattle };

// Declare global 2D array of strings
/*
	A 2D array was created because this connected graph is not directed so if the user wishes
	to figure out the distance a flight travels from LA to Seattle or Seattle to LA, the
string
	will be the same for either.
*/
/*
		Detroit | Miami | Los Angeles | Seattle
	D	--
	M			--
	LA					--
	S							--
*/
const int SIZE = 4;
static std::string distances[SIZE][SIZE] = {
	{ " ", "1153 mi/1855 km", "1983 mi/3192 km", "1938 mi/3119 km" },
	{ "1153 mi/1855 km", " ", "2339 mi/3764 km", "2734 mi/4399 km" },
	{ "1983 mi/3192 km", "2339 mi/3764 km", " ", "959 mi/1544 km" },
	{ "1938 mi/3119 km", "2734 mi/4399 km", "959 mi/1544 km", " " }
};

static void DetroitTo()
{
	// Declare integer variable to hold choice for following switch statement
	int answer;

	// Display menu
	std::cout << "\tArrival to:\n\n";
	std::cout << "1. Miami\n"
		<< "2. Los Angeles\n"
		<< "3. Seattle\n";
	std::cout << "Enter your choice: ";
	std::cin >> answer;

	// validate user input
	while ((answer != 1) && (answer != 2) && (answer != 3))
	{
		std::cout << "Enter a valid number: ";
		std::cin >> answer;
	}

	// Switch statement
	switch (answer)
	{
		case 1:
```

```
		{
			std::cout << "The flight distance from Detroit to Miami is " <<
distances[Detroit][Miami] << std::endl;
			system("PAUSE");
			break;
		}
		case 2:
		{
			std::cout << "The flight distance from Detroit to Los Angeles is " <<
distances[Detroit][LA] << std::endl;
			system("PAUSE");
			break;
		}
		case 3:
		{
			std::cout << "The flight distance from Detroit to Seattle is " <<
distances[Detroit][Seattle] << std::endl;
			system("PAUSE");
			break;
		}
	}
}

static void MiamiTo()
{
	// Declare integer variable to hold choice for following switch statement
	int answer;

	// Display menu
	std::cout << "\tArrival to:\n\n";
	std::cout << "1. Detroit\n"
		<< "2. Los Angeles\n"
		<< "3. Seattle\n";
	std::cout << "Enter your choice: ";
	std::cin >> answer;

	// validate user input
	while ((answer != 1) && (answer != 2) && (answer != 3))
	{
		std::cout << "Enter a valid number: ";
		std::cin >> answer;
	}

	// Switch statement
	switch (answer)
	{
		case 1:
		{
			std::cout << "The flight distance from Miami to Detroit is " <<
distances[Miami][Detroit] << std::endl;
			system("PAUSE");
			break;
		}
		case 2:
		{
			std::cout << "The flight distance from Miami to Los Angeles is " <<
distances[Miami][LA] << std::endl;
			system("PAUSE");
			break;
		}
		case 3:
```

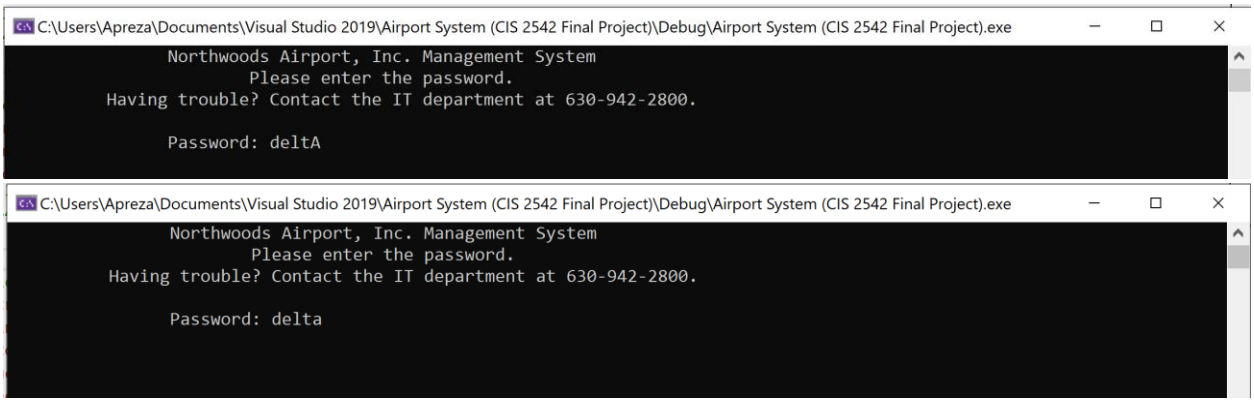
```
		{
			std::cout << "The flight distance from Miami to Seattle is " <<
distances[Miami][Seattle] << std::endl;
			system("PAUSE");
			break;
		}
	}
}

static void LAto()
{
	// Declare integer variable to hold choice for following switch statement
	int answer;

	// Display menu
	std::cout << "\tArrival to:\n\n";
	std::cout << "1. Detroit\n"
		<< "2. Miami\n"
		<< "3. Seattle\n";
	std::cout << "Enter your choice: ";
	std::cin >> answer;

	// validate user input
	while ((answer != 1) && (answer != 2) && (answer != 3))
	{
		std::cout << "Enter a valid number: ";
		std::cin >> answer;
	}

	// Switch statement
	switch (answer)
	{
		case 1:
		{
			std::cout << "The flight distance from Los Angeles to Detroit is " <<
distances[LA][Detroit] << std::endl;
			system("PAUSE");
			break;
		}
		case 2:
		{
			std::cout << "The flight distance from Los Angeles to Miami is " <<
distances[LA][Miami] << std::endl;
			system("PAUSE");
			break;
		}
		case 3:
		{
			std::cout << "The flight distance from Los Angeles to Seattle is " <<
distances[LA][Seattle] << std::endl;
			system("PAUSE");
			break;
		}
	}
}

static void SeattleTo()
{
	// Declare integer variable to hold choice for following switch statement
	int answer;
```

```
	// Display menu
	std::cout << "\tArrival to:\n\n";
	std::cout << "1. Detroit\n"
		<< "2. Miami\n"
		<< "3. Los Angeles\n";
	std::cout << "Enter your choice: ";
	std::cin >> answer;

	// validate user input
	while ((answer != 1) && (answer != 2) && (answer != 3))
	{
		std::cout << "Enter a valid number: ";
		std::cin >> answer;
	}

	// Switch statement
	switch (answer)
	{
		case 1:
		{
			std::cout << "The flight distance from Seattle to Detroit is " <<
distances[Seattle][Detroit] << std::endl;
			system("PAUSE");
			break;
		}
		case 2:
		{
			std::cout << "The flight distance from Seattle to Miami is " <<
distances[Seattle][Miami] << std::endl;
			system("PAUSE");
			break;
		}
		case 3:
		{
			std::cout << "The flight distance from Seattle to Los Angeles is " <<
distances[Seattle][LA] << std::endl;
			system("PAUSE");
			break;
		}
	}
}

/*
	This function allows the user to check the total distance a flight has to travel
	by typing in the number of the place the airplane is departing from. Then, the
appropriate
	function is called where the user has to enter the corresponding number of the place
	the airplane is arriving to.
*/
static void flightDistances()
{
	// Declare integer variable for choice for switch statements
	int choice;

	// Loop to display menu
	do
	{
		system("CLS");
		std::cout << "\tFlight Distances: Departure from:\n\n";
		std::cout << "1. Detroit\n"
			<< "2. Miami\n"
```

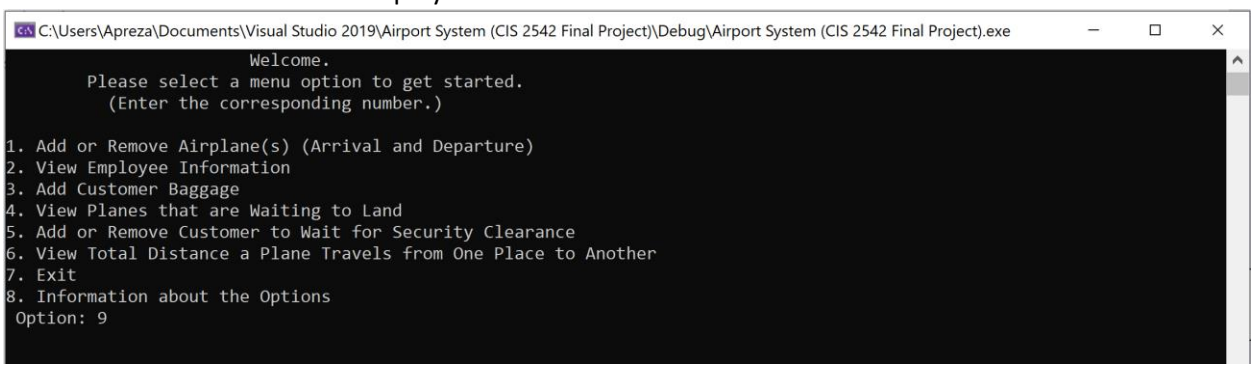
```
                        << "3. Los Angeles\n"
                        << "4. Seattle\n"
                        << "5. Exit\n";
                std::cout << "Enter your choice: ";
                std::cin >> choice;

                switch (choice)
                {
                        case 1:
                                DetroitTo();
                                break;
                        case 2:
                                MiamiTo();
                                break;
                        case 3:
                                LAto();
                                break;
                        case 4:
                                SeattleTo();
                                break;
                        case 5:
                                break;
                        default:
                        {
                                std::cout << "Enter a valid number.\n";
                                system("PAUSE");
                                break;
                        }
                }
        } while (choice != 5);
}
```

Snapshots for Output

Logon

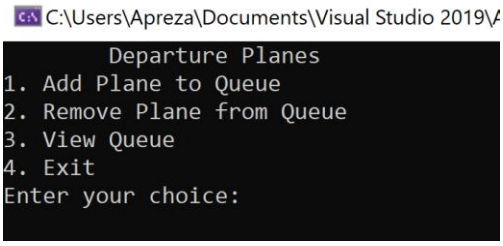


Menu Choices for authorized employee

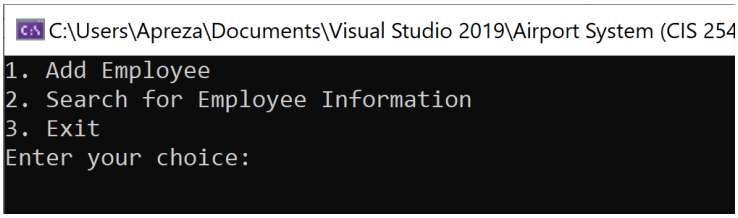


Menu Choice Outputs

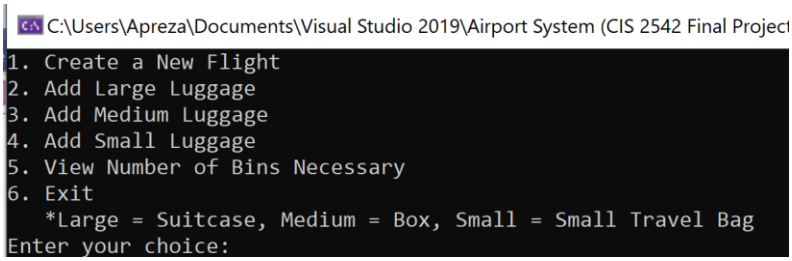
Option 1



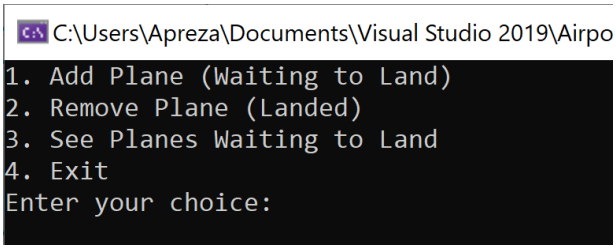
Option 2



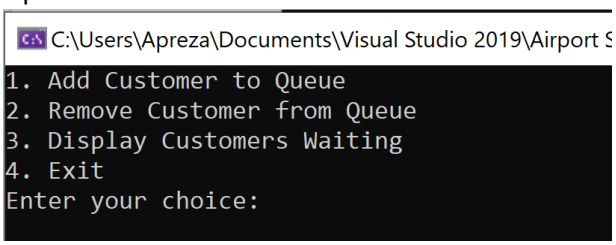
Option 3



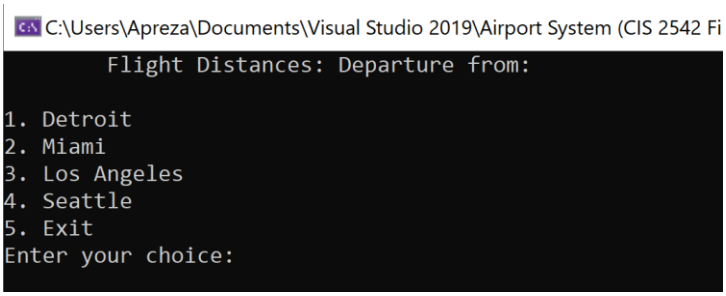
Option 4



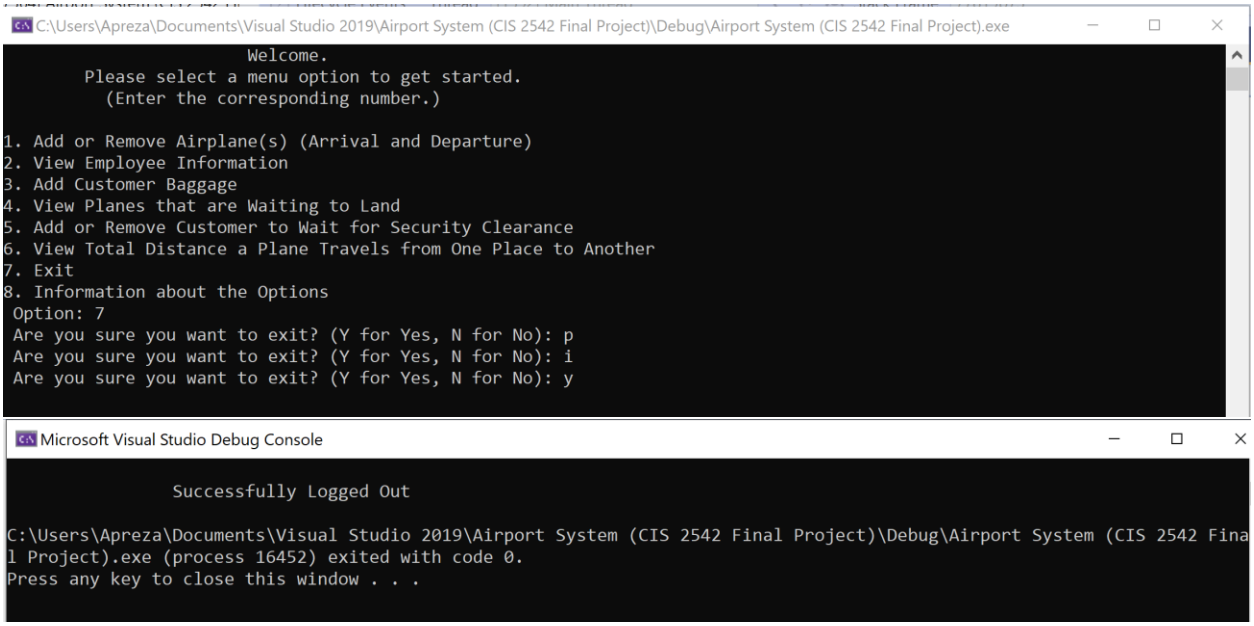
Option 5



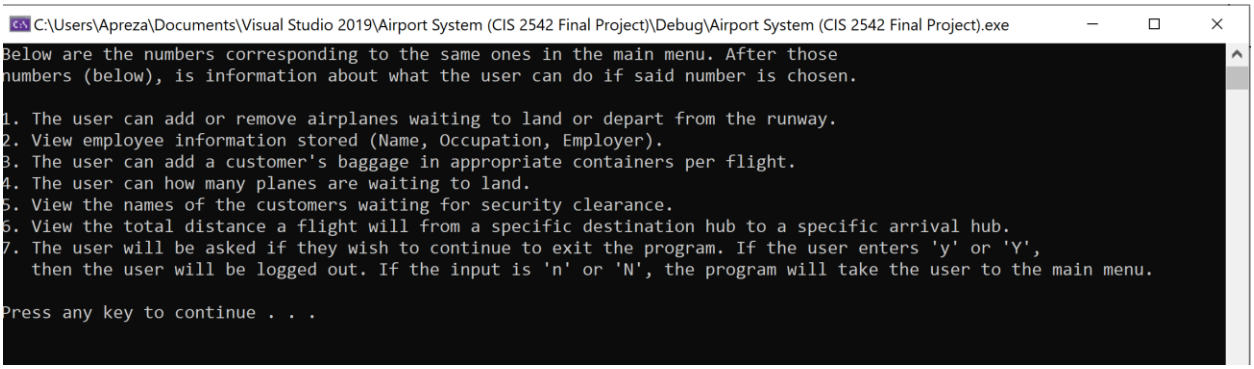
Option 6



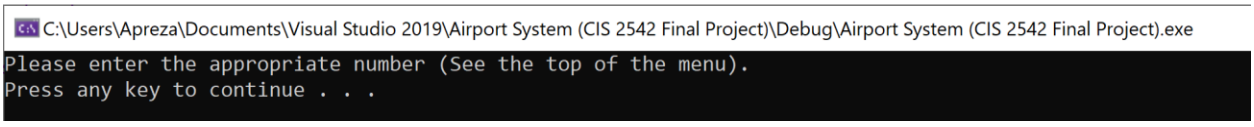
Option 7



Option 8



Default



Menu Choice Output for each of the chosen items.

Option 1

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport
        Departure Planes
1. Add Plane to Queue
2. Remove Plane from Queue
3. View Queue
4. Exit
Enter your choice: 2
There are no planes waiting to depart.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport S
        Departure Planes
1. Add Plane to Queue
2. Remove Plane from Queue
3. View Queue
4. Exit
Enter your choice: 3
There are no planes waiting in queue.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport System
        Departure Planes
1. Add Plane to Queue
2. Remove Plane from Queue
3. View Queue
4. Exit
Enter your choice: 1
Enter the name of the plane waiting for departure: Atlantic Flight 675
Plane successfully added to the queue.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airpor
        Departure Planes
1. Add Plane to Queue
2. Remove Plane from Queue
3. View Queue
4. Exit
Enter your choice: 3

Planes in Queue:
Atlantic Flight 675
Union Flyers 403
Michigan Flight Transport
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System
        Departure Planes
1. Add Plane to Queue
2. Remove Plane from Queue
3. View Queue
4. Exit
Enter your choice: 2
Atlantic Flight 675 has departed.
Union Flyers 403 is next to departure.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport
        Departure Planes
1. Add Plane to Queue
2. Remove Plane from Queue
3. View Queue
4. Exit
Enter your choice: 3

Planes in Queue:
Union Flyers 403
Michigan Flight Transport
Press any key to continue . . .
```

Option 2 (not all possible input and output are shown)

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport
1. Add Employee
2. Search for Employee Information
3. Exit
Enter your choice: 2
There are no employees on record.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Fin
Enter the employee's name: John Smith
Enter the employee's occupation: Pilot
Enter the employee's employer: Johnson Airlines
Employee successfully added.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (C
Enter the employee's name: Michelle Apreza
Enter the employee's occupation: IT
Enter the employee's employer: COD
Employee successfully added.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport Sy
Will you be searching for the employee by name, occupation, or employer?
Enter N for Name, O for occupation, or E for Employer: n
Enter the name of the employee: John Smith
Employee Name: John Smith
Employee Occupation: Pilot
Employee Employer: Johnson Airlines
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport System (CIS 2542 Fina
Will you be searching for the employee by name, occupation, or employer?
Enter N for Name, O for occupation, or E for Employer: o
Enter the occupation of the employee: Mechanic
Employee Name: Bob Sands
Employee Occupation: Mechanic
Employee Employer: Johnson Airlines
Press any key to continue . . .
Would you like to search for another employee? (Enter any letter for Yes, N for No): y
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport System (CIS 2542
Will you be searching for the employee by name, occupation, or employer?
Enter N for Name, O for occupation, or E for Employer: e
Enter the employer of the employee: NorthWoods
Employee was not found. Please try again.
Would you like to search for another employee? (Enter any letter for Yes, N for No):
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport System (CIS 2542 Final
Will you be searching for the employee by name, occupation, or employer?
Enter N for Name, O for occupation, or E for Employer: E
Enter the employer of the employee: Northwoods
Employee Name: Amelia Hill
Employee Occupation: Airport Security Mgr.
Employee Employer: Northwoods
Press any key to continue . . .
Would you like to search for another employee? (Enter any letter for Yes, N for No): N
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport System (CIS 2
Will you be searching for the employee by name, occupation, or employer?
Enter N for Name, O for occupation, or E for Employer: n
Enter the name of the employee: Michelle Apreza
Employee Name: Michelle Apreza
Employee Occupation: IT
Employee Employer: COD
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport System (CIS 2542 F
Will you be searching for the employee by name, occupation, or employer?
Enter N for Name, O for occupation, or E for Employer: o
Enter the occupation of the employee: IT
Employee Name: Michelle Apreza
Employee Occupation: IT
Employee Employer: COD
Press any key to continue . . .
Would you like to search for another employee? (Enter any letter for Yes, N for No): y
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport System (CIS 2542 Final
Will you be searching for the employee by name, occupation, or employer?
Enter N for Name, O for occupation, or E for Employer: e
Enter the employer of the employee: COD
Employee Name: Michelle Apreza
Employee Occupation: IT
Employee Employer: COD
Press any key to continue . . .
Would you like to search for another employee? (Enter any letter for Yes, N for No): y
```

Option 3

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airpor
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
   *Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 2
Please create a flight (Option 1).
Press any key to continue . . .
```

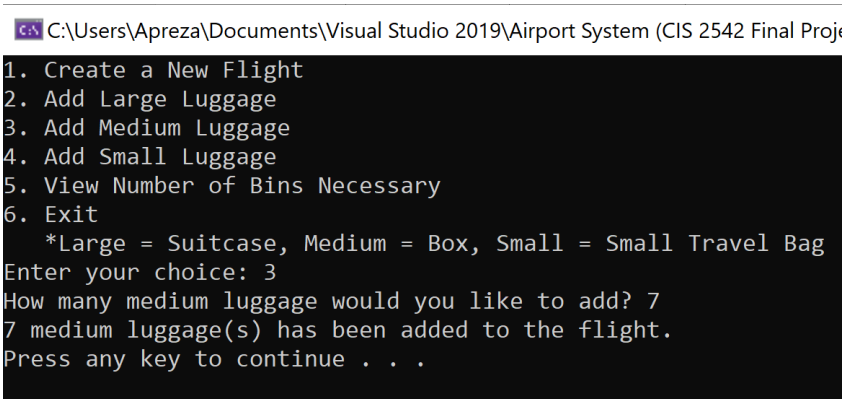
```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Proje
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
   *Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 3
Please create a flight (Option 1).
Press any key to continue . . .
```

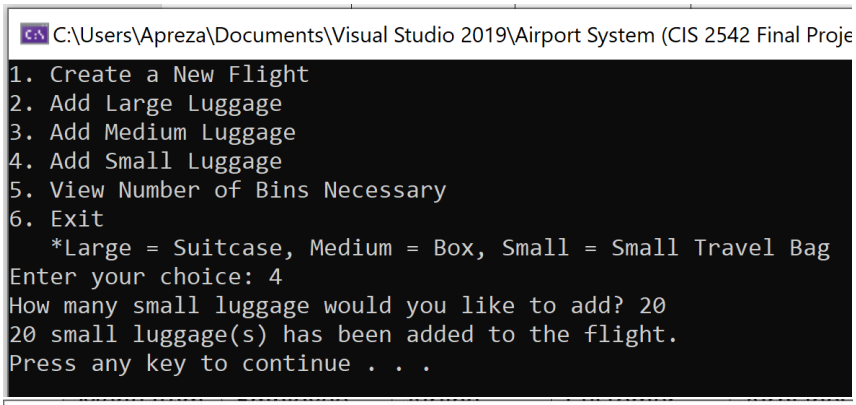
```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Proje
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
   *Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 4
Please create a flight (Option 1).
Press any key to continue . . .
```

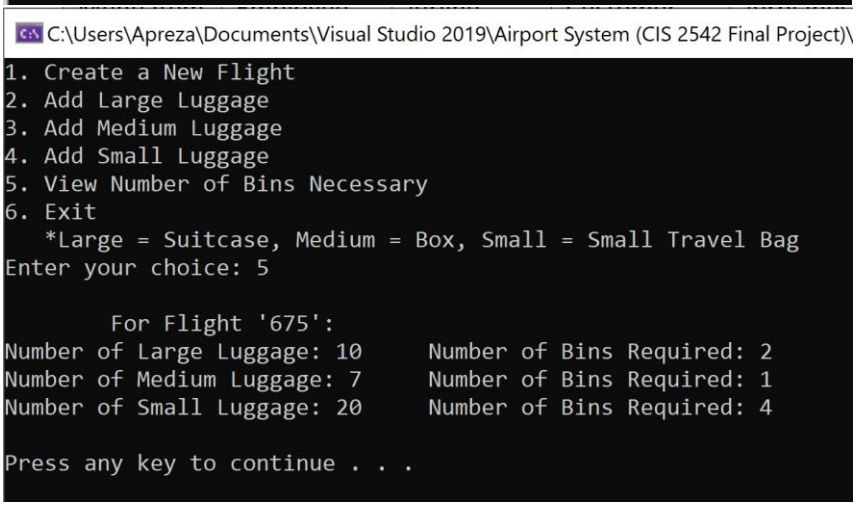
```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Proj
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
   *Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 5
Please create a flight (Option 1).
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Proje
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
   *Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 1
Enter the flight number: 675
Flight successfully created.
Press any key to continue . . .
```

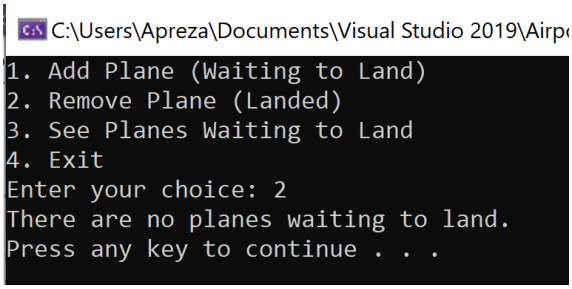
```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Proje
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
   *Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 2
How many large luggage would you like to add? 10
10 large luggage(s) has been added to the flight.
Press any key to continue . . .
```


C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Proje
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
*Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 3
How many medium luggage would you like to add? 7
7 medium luggage(s) has been added to the flight.
Press any key to continue . . .


C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Proje
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
*Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 4
How many small luggage would you like to add? 20
20 small luggage(s) has been added to the flight.
Press any key to continue . . .


C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\
1. Create a New Flight
2. Add Large Luggage
3. Add Medium Luggage
4. Add Small Luggage
5. View Number of Bins Necessary
6. Exit
*Large = Suitcase, Medium = Box, Small = Small Travel Bag
Enter your choice: 5
For Flight '675':
Number of Large Luggage: 10 Number of Bins Required: 2
Number of Medium Luggage: 7 Number of Bins Required: 1
Number of Small Luggage: 20 Number of Bins Required: 4
Press any key to continue . . .

Option 4


C:\Users\Apreza\Documents\Visual Studio 2019\Airp
1. Add Plane (Waiting to Land)
2. Remove Plane (Landed)
3. See Planes Waiting to Land
4. Exit
Enter your choice: 2
There are no planes waiting to land.
Press any key to continue . . .

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airpo
1. Add Plane (Waiting to Land)
2. Remove Plane (Landed)
3. See Planes Waiting to Land
4. Exit
Enter your choice: 3
There are no planes waiting to land.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport S
1. Add Plane (Waiting to Land)
2. Remove Plane (Landed)
3. See Planes Waiting to Land
4. Exit
Enter your choice: 1
Enter the name of the plane waiting to land: Pennsylvania Air Flight 987
Plane successfully added to the stack.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport
1. Add Plane (Waiting to Land)
2. Remove Plane (Landed)
3. See Planes Waiting to Land
4. Exit
Enter your choice: 3

Planes waiting to land:
Jaguar Airlines Flight 221
TransOceanic Flight 753
Atlantic Flight 345
Pennsylvania Air Flight 987
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport S
1. Add Plane (Waiting to Land)
2. Remove Plane (Landed)
3. See Planes Waiting to Land
4. Exit
Enter your choice: 2
Jaguar Airlines Flight 221 has landed.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\A
1. Add Plane (Waiting to Land)
2. Remove Plane (Landed)
3. See Planes Waiting to Land
4. Exit
Enter your choice: 3

Planes waiting to land:
TransOceanic Flight 753
Atlantic Flight 345
Pennsylvania Air Flight 987
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airpo
1. Add Plane (Waiting to Land)
2. Remove Plane (Landed)
3. See Planes Waiting to Land
4. Exit
Enter your choice: 2
TransOceanic Flight 753 has landed.
Press any key to continue . . .
```

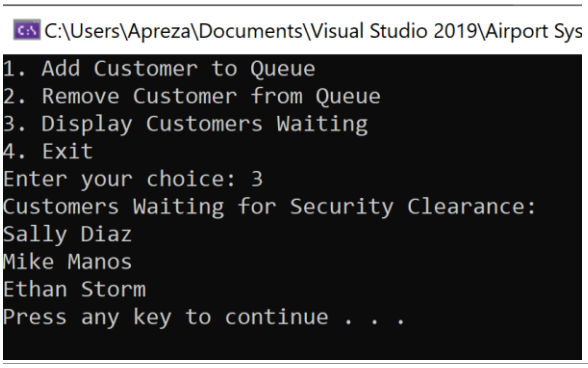
```
C:\Users\Apreza\Documents\Visual Studio 2019\Airp
1. Add Plane (Waiting to Land)
2. Remove Plane (Landed)
3. See Planes Waiting to Land
4. Exit
Enter your choice: 3

Planes waiting to land:
Atlantic Flight 345
Pennsylvania Air Flight 987
Press any key to continue . . .
```

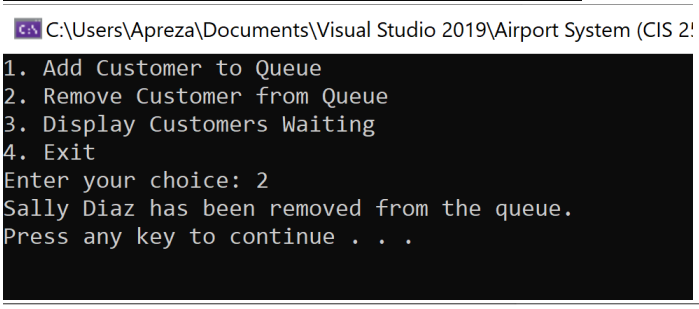
Option 5

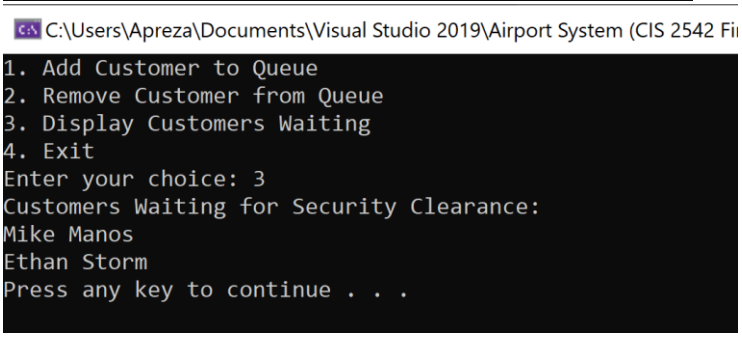
```
1. Add Customer to Queue
2. Remove Customer from Queue
3. Display Customers Waiting
4. Exit
Enter your choice: 2
There are no customers in line.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Air
1. Add Customer to Queue
2. Remove Customer from Queue
3. Display Customers Waiting
4. Exit
Enter your choice: 3
There are no customers in line.
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport System
1. Add Customer to Queue
2. Remove Customer from Queue
3. Display Customers Waiting
4. Exit
Enter your choice: 1
Enter the name of the customer to waiting for security clearance: Sally Diaz
Customer successfully added to the queue.
Press any key to continue . . .
```


C:\Users\Apreza\Documents\Visual Studio 2019\Airport Sys
1. Add Customer to Queue
2. Remove Customer from Queue
3. Display Customers Waiting
4. Exit
Enter your choice: 3
Customers Waiting for Security Clearance:
Sally Diaz
Mike Manos
Ethan Storm
Press any key to continue . . .


C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2
1. Add Customer to Queue
2. Remove Customer from Queue
3. Display Customers Waiting
4. Exit
Enter your choice: 2
Sally Diaz has been removed from the queue.
Press any key to continue . . .


C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Fi
1. Add Customer to Queue
2. Remove Customer from Queue
3. Display Customers Waiting
4. Exit
Enter your choice: 3
Customers Waiting for Security Clearance:
Mike Manos
Ethan Storm
Press any key to continue . . .

Option 6 (Not all possible routes were shown)

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airp
        Flight Distances: Departure from:

1. Detroit
2. Miami
3. Los Angeles
4. Seattle
5. Exit
Enter your choice: 1
        Arrival to:

1. Miami
2. Los Angeles
3. Seattle
Enter your choice: 1
The flight distance from Detroit to Miami is 1153 mi/1855 km
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Air
        Flight Distances: Departure from:

1. Detroit
2. Miami
3. Los Angeles
4. Seattle
5. Exit
Enter your choice: 2
        Arrival to:

1. Detroit
2. Los Angeles
3. Seattle
Enter your choice: 1
The flight distance from Miami to Detroit is 1153 mi/1855 km
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport
        Flight Distances: Departure from:

1. Detroit
2. Miami
3. Los Angeles
4. Seattle
5. Exit
Enter your choice: 3
        Arrival to:

1. Detroit
2. Miami
3. Seattle
Enter your choice: 3
The flight distance from Los Angeles to Seattle is 959 mi/1544 km
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airpor
        Flight Distances: Departure from:

1. Detroit
2. Miami
3. Los Angeles
4. Seattle
5. Exit
Enter your choice: 4
        Arrival to:

1. Detroit
2. Miami
3. Los Angeles
Enter your choice: 3
The flight distance from Seattle to Los Angeles is 959 mi/1544 km
Press any key to continue . . .
```

C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airport Sy

```
       Flight Distances: Departure from:

1. Detroit
2. Miami
3. Los Angeles
4. Seattle
5. Exit
Enter your choice: 1
       Arrival to:

1. Miami
2. Los Angeles
3. Seattle
Enter your choice: 2
The flight distance from Detroit to Los Angeles is 1983 mi/3192 km
Press any key to continue . . .
```

C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airpor

```
       Flight Distances: Departure from:

1. Detroit
2. Miami
3. Los Angeles
4. Seattle
5. Exit
Enter your choice: 1
       Arrival to:

1. Miami
2. Los Angeles
3. Seattle
Enter your choice: 4
Enter a valid number: 3
The flight distance from Detroit to Seattle is 1938 mi/3119 km
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\Airpor
        Flight Distances: Departure from:

1. Detroit
2. Miami
3. Los Angeles
4. Seattle
5. Exit
Enter your choice: 2
        Arrival to:

1. Detroit
2. Los Angeles
3. Seattle
Enter your choice: 2
The flight distance from Miami to Los Angeles is 2339 mi/3764 km
Press any key to continue . . .
```

```
C:\Users\Apreza\Documents\Visual Studio 2019\Airport System (CIS 2542 Final Project)\Debug\A
        Flight Distances: Departure from:

1. Detroit
2. Miami
3. Los Angeles
4. Seattle
5. Exit
Enter your choice: 2
        Arrival to:

1. Detroit
2. Los Angeles
3. Seattle
Enter your choice: 3
The flight distance from Miami to Seattle is 2734 mi/4399 km
Press any key to continue . . .
```

Conclusion

During this process, my perception of data structures changed. I previously thought that when it came to applying data structures, I would have to literally go through many, or even all, data structures to decide which one(s) would fit into the creation of the software. I was exposed to many types of data structures and somehow, I knew which one I could apply and how.

I used the STL for the vectors, queues, and stacks, in which there is no shame to use. It does save resources. I have already used the vector and stack libraries, but I have never used the queue library. However, I have created a queue implemented by an array, but I still knew how a queue worked. The struct was used for employee records because there was not much information the struct would hold. Had there been a need for function to manipulate that data, along with an increase is data, a class would have been a better option. Another thing that surprised me was my emphasis on using pointers. I knew that they were extremely helpful, but I learned that instead of passing data back and forth between functions, a pointer could be sent to a function and it would not need to return. I used to not like pointer much, but after this project, I gained an appreciation for them. Another point I would like to emphasize is how I implemented a connected graph. Like any other data structure, it could be implemented multiple ways, and I initially thought that those ways were difficult. In this project, I used a two-dimensional array. Because I used a non-directed connected graph, more ‘cells’ could be filled to simulate the fact that the graph is not directed.

I would have focused on making the program as short as possible to occupy less memory but remembered the fact that it still must be simple for any user to use, but tried to keep the previous factor in mind nonetheless. Therefore, the lesson learned here is that there must be compromise, perhaps multiple times, any type of software is created. After testing the program, I can conclude that it is easy for a veteran user to navigate and still just about easy enough for anyone else who is using for the first time, which is also why I added an option in the main menu where the user can read about what each of the functions do.

References

Airport Technology Management: Operations, Software Solutions and Vendors. (2018, October 15). Retrieved December 8, 2019, from https://www.altexsoft.com/blog/travel/airport-technology-management-operations-software-solutions-and-vendors/.

Distance Calculator. Retrieved December 11, 2019, from https://www.travelmath.com/distance/.