University of Windsor

Fall 2023: Midterm Exam

COMP-1410-1 Introduction to Algorithms & Programming II

Thursday, November 16, 2023

School of Computer Science
University of Windsor

READ THE FOLLOWING INSTRUCTION BEFORE GOING TO THE QUESTIONS

- The time limit is 80 minutes.
- The exam is out of 60 marks.
- For every question:
  - Thoroughly and carefully read the description, comments for each incomplete function or partial code that you should complete, and the sample execution code before start writing your answers.
  - Complete the code
  - **DO NOT ALTER** any existing code.
  - Do not use any non-standard library
  - The spaces provided in every incomplete code for you to complete it is more than enough.

This is a "**closed book**" exam. No reference material, calculators or any electronic equipment is permitted.

Student Name	
Student ID	

Please also write your name and student ID on top of every page.

Student's Signature: _____

GOOD LUCK!

Student Name: Student ID:

Question	1	2	Bonus	Total
Total Mark	30	30	6	60
Your Mark				

Question 1 (30 marks)

Program: Fancy car

Complete the c program below that uses a `struct` to support basic operations **drive**, **add fuel**, **start**, and **stop engine**. The program includes three files, **FancyCar.h**, **FancyCar.c**, and **MyCar.c**.

- **FancyCar.h**: Declares the `FancyCar_Struct`, and the prototypes of five functions. You have already been provided with the function prototypes.
  Your Tasks: Complete the struct type `FancyCar` as follows:
  Data members: car `model` (`char[]`) (already declared), kpl (Kilometers per Liter) (double), fuel capacity (double), amount of fuel in tank (double), odometer (int), and engine status (`int`).

- **FancyCar.c**: It should have the implementation of every function defined in its corresponding header file.
  Your Tasks: Complete the following five functions:

  - `InitializeFancyCar()` to set the car model, Kilometers per Liter (KPL), fuel capacity, fuel in tank, and odometer with their corresponding arguments. `InitializeFancyCar()` returns the initialized `FancyCar struct`. Initialize with the engine off (`0`).

  - `Drive()`: This function only updates data members **if the engine is on**. Kilometers driven should increase by parameter `distance`, and the amount of fuel should decrease by `distance / kpl`. **Data members are only updated if parameter `distance` is positive**. If the car runs out of fuel, the parameter distance will not be achieved, and the fule tank will have 0.0 liters. For example, Drive(car, 100) will not be possible with only three liters of fuel and KPL of 20.0. The maximum driving distance is 60 kilometers with three liters of fuel and KPL of 20.0. Therefore, the odometer will only increase by 60 instead of the requested 100, and the fuel tank will have 0.0 liters (not a negative amount). The engine turns off if the car runs out of fuel. `Drive()` method returns the updated `FancyCar struct`.

  - `AddFuel()`: Only add fuel if the engine is off. Increase fuel tank by parameter `amount` only if parameter is positive. Set fuel tank to full if too much fuel was added. `AddFuel()` returns the updated `FancyCar struct`.

  - `StartEngine()`: to set the engine status to `1` and return the updated `FancyCar struct` variable.

  - `StopEngine()`: to set the engine status to `0` and return the updated `FancyCar struct` variable.

- **MyCar.c**: Using the comments and print commands in the `main` function and the execution outputs provided
  - Complete the `main` function.
  - Add one include statement on top of **MyCar.c** to use the `FancyCar` custom library.

Execution output of `MyCar.c`**:**

```
A Honda car is initialized, with KPL=9.5, Fuel Capacity=50, Fuel Level=20, and Odometer=1000.
Car engine is Off
Car tries to drive 10 kilometers when its engine is
off! Current Odometer: 1000
Car engine is now On
Car drives 100 kilometers when its engine is on.
Current Odometer: 1100
Amount of fuel in tank:  9.47
Car tries to drive 200 more kilometers.
Current Odometer: 1190
Amount of fuel in tank:  0.00
Fill 60 liters of fuel.
Amount of fuel in tank: 50.00
Car tries to drive 200 more kilometers.
Current Odometer: 1390
Amount of fuel in tank: 28.95
Car engine is now Off
```

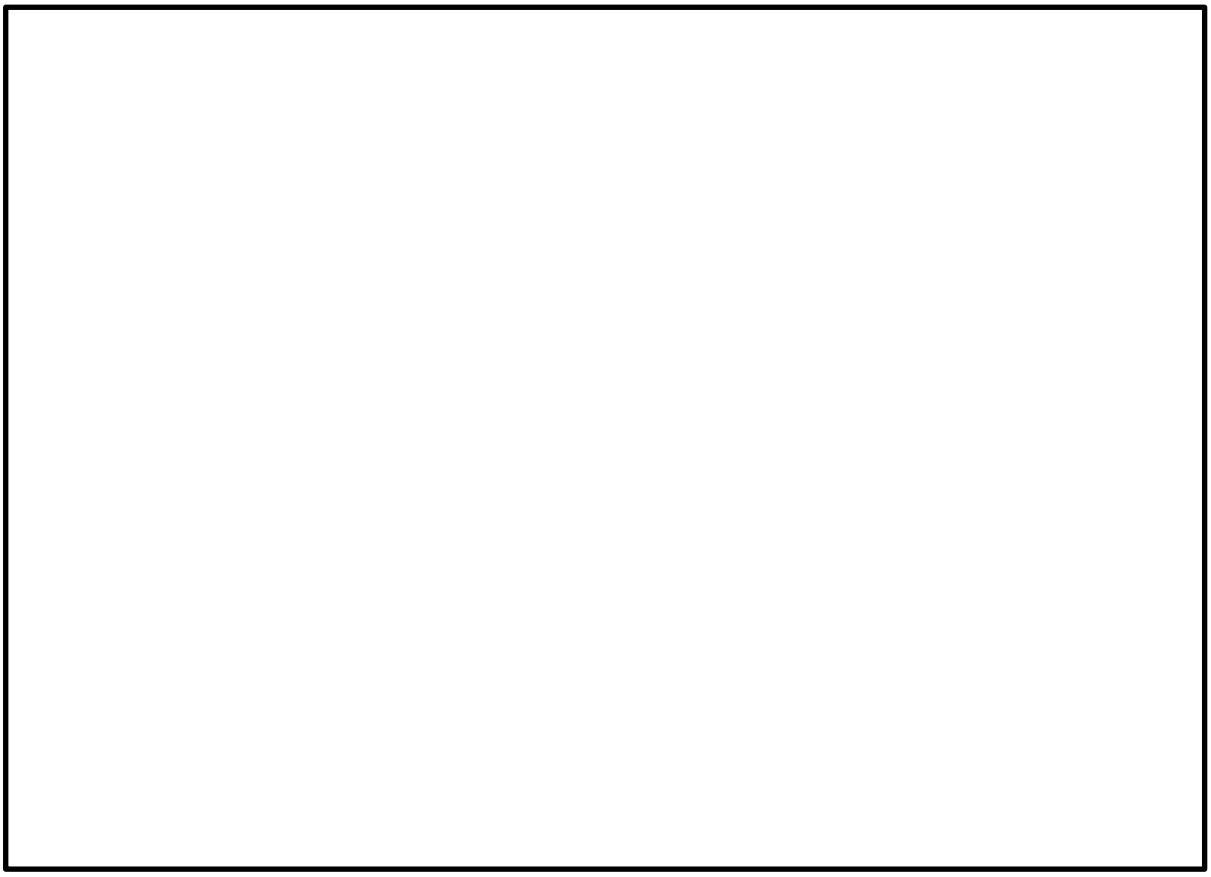
FancyCar.h **(5 marks)**

```
#ifndef FANCY_CAR_H
#define FANCY_CAR_H

#include <stdio.h>
#include <string.h>

// Complete the struct type, FancyCar, with required members                    (5 marks)

typedef struct FancyCar_Struct {
        char model[100];
} FancyCar;

// Function Prototypes.
// No action needed here. These functions must be implemented
// in the corresponding c file, FancyCar.c.

// Car initialization
FancyCar InitializeFancyCar(FancyCar car, char* model, double kpl,
                                double fuelCapacity, double fuel, int odometer);

// Drive car requested kilometers but check for enough
// fuel and check for positive value
FancyCar Drive(FancyCar car, int kilometersToDrive);

// Add fuel to tank. Check for positive value.
FancyCar AddFuel(FancyCar car, int amtToAdd);

// Set engine status to 1
FancyCar StartEngine(FancyCar car);

// Set engine status to 0
FancyCar StopEngine(FancyCar car);

#endif
```

FancyCar.c **(16 marks)**

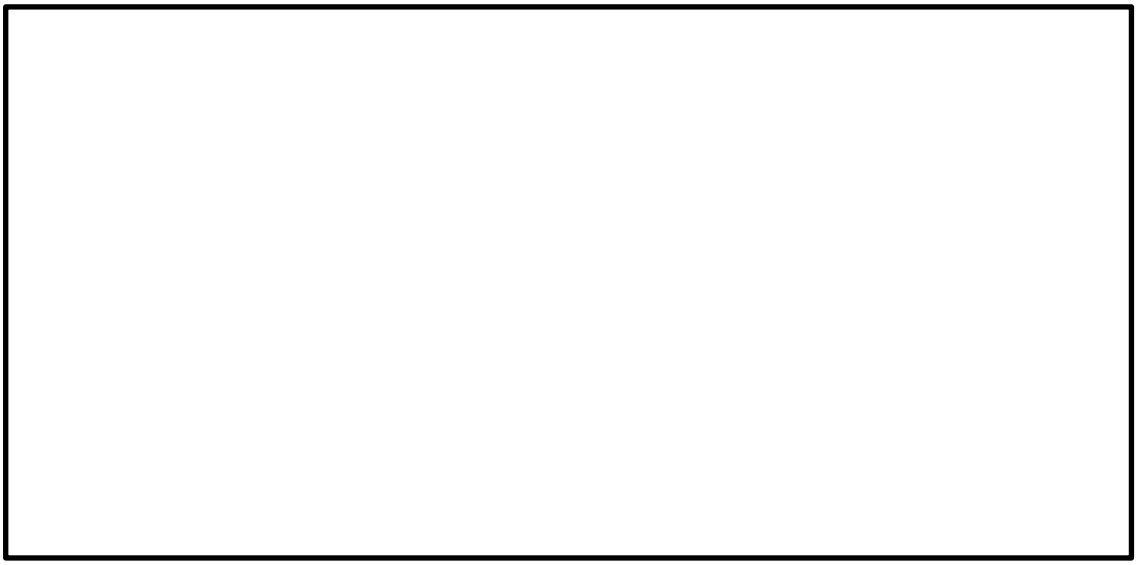
```
#include <stdio.h>
#include <string.h>
#include "FancyCar.h"

// Car initialization
FancyCar InitializeFancyCar(FancyCar car, char* model, double kpl,
                            double fuelCapacity,double fuel, int odometer) {
   // Complete the function body.                                          (3 marks)
```

```
}

// Drive car requested kilometers
// It must check for engine status, enough fuel and check for positive
value FancyCar Drive(FancyCar car, int kilometersToDrive) {

   // Complete the function body.                                          (7 marks)
```



```
}
```

```
// Add fuel to tank
// It must Check for engine status, fuel capacity and positive value.
FancyCar AddFuel(FancyCar car, int amtToAdd) {
   // Complete the function body.                                    (4 marks)
```



```
}

// Turn on the car engine
FancyCar StartEngine(FancyCar car) {

   // Complete the function body.                                    (1 mark)
```

```
}

// Turn off the car engine
FancyCar StopEngine(FancyCar car) {

   // Complete the function body.                                    (1 mark)
```

```
}
```

MyCar.c **(9 marks)**

```
#include <stdio.h>
#include <string.h>
// Include FancyCar library                                                    (0.5 mark)

int main(void) {
   // Declare a FancyCar struct variable                                       (0.5 mark)

   _____
   puts("A Honda, with KPL=9.5, Fuel Capacity=50, Fuel Level=20, and Odometer=1000.");
   // Initialize the car with above information.                               (2 marks)

   _____
   //Show the car engine status (on or off)                                    (0.5 mark)

   _____
   puts("Car tries to drive 10 kilometers when its engine is off!");
   // Drive the car for 10 kilometers.                                         (0.5 mark)

   _____
   printf("Current Odometer: %d\n",car.odometer);

   // Turn of the car engine.                                                  (0.5 mark)

   _____
   //Show the car engine status (on or off)                                    (0.5 mark)

   _____
   puts("Car drives 100 kms when its engine is on.");
   // Drive the car for 100 kilometers.                                        (0.5 mark)

   _____
   printf("Current Odometer:  %d\n",car.odometer);
   // Show the remaining fuel in trunk.                                        (0.5 mark)

   _____
   puts("Car tries to drive 200 more kilometers.");
   // Drive the car for 100 kilometers.                                        (0.5 mark)

   _____
   printf("Current Odometer: %d\n",car.odometer);
   printf("Amount of fuel in tank: %5.2f\n",car.fuelTank);
   puts("Fill 60 liters of fuel.");
   // Fill 60 liters of fuel.                                                  (0.5 mark)

   _____
   printf("Amount of fuel in tank: %5.2f\n",car.fuelTank);
   // Drive the car for 200 kilometers.                                        (1 mark)
   puts("Car tries to drive 200 more kilometers.");


   _____


   _____
   printf("Current Odometer: %d\n",car.odometer);
   printf("Amount of fuel in tank: %5.2f\n",car.fuelTank);
   // Turn off the car engine                                                  (0.5 mark)

   _____
   //Show the car engine status (on or off)                                    (0.5 mark)


   _____
   return 0;
}
```

Extra page, if needed, for Question 1

Question 2 (30 marks)

Program: Seat Reservation

Complete the incomplete c program below which has some simple operations for a seat reservation system, in the following steps:

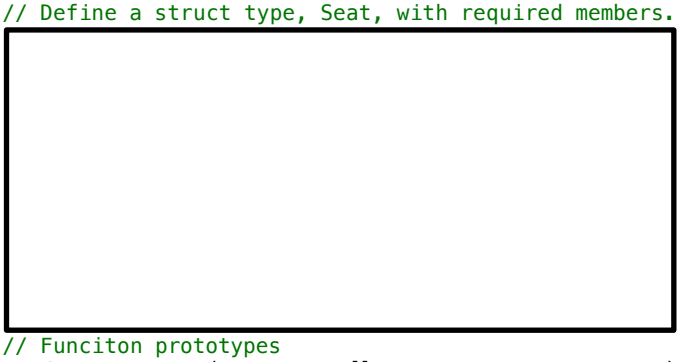
- Define a struct type, `Seat`, with the following members:
    - Two `char` arrays with the length of `50` each to keep `firstName` and `lastName`.
    - One integer, `amountPaid`.
- In the `main` function, complete the four parts that have indicated by the corresponding comments. Note that in each part you only need to write one statement, which could be a function call with the correct argument(s).
- Write the body of the following six functions. Note that the prototypes of these functions have already been provided on top of the program, and you only need to write their bodies after the `main` function.
    - `ReserveSeat`: Reserves a `seat` using the `seat` array, `seat` number and a `seat` record.
    - `MakeSeatEmpty`: It makes one `seat` empty using a pointer `seat` record.
    - `MakeAllEmpty`: It makes all the seats empty by repeatedly calling the function `MakeSeatEmpty`.
    - `IsSeatEmpty`: It checks if a given `seat` is empty or not, using a `seat` record.
    - `PrintSeat`: It prints one `seat` using a `seat` record.
    - `PrintAll`: It prints all the seats by repeatedly calling the function `PrintSeat`.

Execution output of `SeatReservation.c`**:**

```
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): p
1: empty empty, Paid: 0
2: empty empty, Paid: 0
3: empty empty, Paid: 0
4: empty empty, Paid: 0
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): r
Enter seat num: 0
Invalid seat number!
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): r
Enter seat num: 3
Enter first name:
Hilary
Enter last name:
Clinton
Enter amount paid:
1400
Seat Reserved.
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): r
Enter seat num: 3
Seat has already been reserved by someone.
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): r
Enter seat num: 4
Enter first name:
Bill
Enter last name:
Clinton
Enter amount paid:
1350
Seat Reserved.
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): p
1: empty empty, Paid: 0
2: empty empty, Paid: 0
3: Hilary Clinton, Paid: 1400
4: Bill Clinton, Paid: 1350
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): e
Enter seat num:
1
Seat is already empty.
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): e
Enter seat num:
3
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): p
1: empty empty, Paid: 0
2: empty empty, Paid: 0
3: empty empty, Paid: 0
4: Bill Clinton, Paid: 1350
Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): q
Goodbye...!
```

SeatReservation.c **(30 marks)**

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdbool.h>
// Define a struct type, Seat, with required members.              (4 marks)
```



```
// Funciton prototypes
void ReserveSeat(Seat seats[],int seatNum, Seat seat);
void MakeSeatEmpty(Seat* seat);
void MakeAllEmpty(Seat seats[], int numSeats);
bool IsSeatEmpty(Seat seat);
void PrintSeat(Seat seat);
void PrintAll(Seat seats[], int numSeats);

int main(void) {
   const int NUM_SEATS = 5;
   char userKey;
   int  seatNum;
   Seat allSeats[NUM_SEATS];
   Seat currSeat;
   userKey = '-';
   seatNum = 0;
   // Empty all the seats                                          (2 marks)

   ____________________________________________________________

   while (toupper(userKey) != 'Q') {
      printf("Enter command ((P)rint/(R)eserve/(E)mpty/(Q)uit): \n");
      scanf(" %c", &userKey);
      if (toupper(userKey) == 'P') {
         // Print all the seats                                    (2 marks)

         ____________________________________________________________
      }
      else if (toupper(userKey) == 'R') {
         printf("Enter seat num: \n");
         scanf("%d", &seatNum);
         if (seatNum <=0 || seatNum>NUM_SEATS)
            printf("Invalid seat number!\n");
         else if (!IsSeatEmpty(allSeats[seatNum-1]))
            printf("Seat has already been reserved by someone.\n\n");
         else {
            printf("Enter first name: \n");
            scanf("%s", currSeat.firstName);
            printf("Enter last name: \n");
            scanf("%s", currSeat.lastName);
            printf("Enter amount paid: \n");
            scanf("%d", &currSeat.amountPaid);
            // Reserve the seat                                    (2 marks)

            ____________________________________________________________
            printf("Seat Reserved.\n\n");
         }
      }
```

```
      else if (toupper(userKey) == 'E') {
         printf("Enter seat num: \n");
         scanf("%d", &seatNum);
          if (IsSeatEmpty(allSeats[seatNum-1]))
            printf("Seat is already empty.\n\n");
         else if (seatNum >0 && seatNum<=NUM_SEATS)
            // Make the selected seat empty                    (2 marks)


            _____________________________________________________________

         else
            printf("Invalid seat number!\n");
      }
      else if (toupper(userKey) == 'Q')
         printf("Goodbye...!\n");
      else
         printf("Invalid command.\n\n");
   }
   return 0;
}
// Implement the functions below.

// Reserve a seat                                          (2 marks)
void ReserveSeat(Seat seats[],int seatNum, Seat seat) {


     ____________________________________
}
// Make one seat empty                                     (4 marks)
void MakeSeatEmpty(Seat* seat) {
     [                                  ]
     [                                  ]
     [                                  ]
     [                                  ]
}
// Make all seats empty using MakeSeatEmpty function       (3 marks)
void MakeAllEmpty(Seat seats[], int numSeats) {
     [                                  ]
     [                                  ]
     [                                  ]
     [                                  ]
}
// Check if a seat is  empty                               (2 marks)
bool IsSeatEmpty(Seat seat) {


     ____________________________________
}
// Print a seat                                            (4 marks)
void PrintSeat(Seat seat) {
     [                                  ]
     [                                  ]
     [                                  ]
     [                                  ]
}
// Print all seats using PrintSeat function                (3 marks)
void PrintAll(Seat seats[], int numSeats) {
     [                                  ]
     [                                  ]
     [                                  ]
     [                                  ]
}
```

Extra page, if needed, for Question 2

Bonus Question (6 marks)

Frequency of the words with different lengths

Using the comments inside the incomplete c program below, complete the program to calculate and show the **total number of words** and **frequency of words with different lengths** in a string.

The execution output of the complete program has been provided on the next page.

```
#include <stdio.h>
// Include the library that has the strtok function in it.         (0.5 mark)

____________________

int main(void) {
	// The text content
	char line[] = "In this c program, we count and show the frequency of words with different lengths.";
	printf("Sentence: \"%s\"\n", line);

	// Declare and initialize an array, wordLength, to keep the word's lengths
	// In the first element of the array, store the total number of words.
	// In each array element with index i, store the count of words with length i.

	int wordLength_______________;                                                    (0.5 mark)

	// Start tokenizing the array line using delimiters " .,?!:;"

	char *tokenPtr = ___________________________________;                             (0.5 mark)

	// In a while loop, for each token

	while (____________________) {                                                    (0.5 mark)

		// Update the total number of words in the first element of the array

		_____________________;                                                       (0.5 mark)

		// Check the length of the token.
		// Update the corresponding element of the array wordLength.
		// *Note: If the length of the token is equal to or greater than 7,
		// you should update the last element of the array wordLength.

		if (_____________________________)                                            (0.5 mark)


			_____________________________;                                       (0.5 mark)
		else

			_____________________________;                                       (0.5 mark)

		// Then get the next token.

		_____________________________;                                               (0.5 mark)

	}
	// Print the total number of words
	printf("Total Number of Words: %d.\n", ____________________);                    (0.5 mark)

	// Print out the word frequency table
	printf("Word Length\tFrequency\n========================\n");

	for (int i = 1; i <= 7; i++) {
		if (i < 7)
			printf("%d\t\t%3d\n",i, ____________________);                       (0.5 mark)
		else
			printf(">=7\t\t%3d\n",____________________);                         (0.5 mark)
	}

	puts("========================");

}
```

Execution output:

```
Sentence: "In this c program, we count and show the frequency of words with different lengths."
Total Number of Words: 15.
Word Length     Frequency
========================
1                  1
2                  3
3                  2
4                  3
5                  2
6                  0
>=7                4
========================
```