De La Salle University
College of Computer Studies
Department of Software Technology

CCPROG2 Machine Project 2T AY2021-2022
Arrows Express Line Embarkation System

De La Salle University has a shuttle service called the Arrows Express Line that travels to and from the Laguna campus on a daily basis that Faculty, Staff, Researchers and Students can avail of. The service has a fixed schedule each day as follows:

Trip Number	ETD	Route
AE101	6:00 am	Manila to Laguna
AE102	7:30 am	
AE103	9:30 am	
AE104	11:00 am	
AE105	1:00 pm	
AE106	2:30 pm	
AE107	3: 30 pm	
AE108	5:00 pm	
AE109	6:15 pm	
AE150	5:30 am	Laguna to Manila
AE151	5:45 am	
AE152	7:00 am	
AE153	7:30 am	
AE154	9:00 am	
AE155	11:00 am	
AE156	1:00 pm	
AE157	2:30 pm	
AE158	3:30 pm	
AE159	5:00 pm	
AE160	6:15 pm	

The following are the embarkation points and drop-off points of the service:

Embarkation Points	Drop-Off Points
DLSU Manila Campus - South Gate Driveway	• ***Via Mamplasan Exit*:** 1st drop-off point – Mamplasan Toll Exit 2nd drop-off point – Phase 5, San Jose Village 3rd drop-off point – Milagros Del Rosario (MRR) Building – East Canopy • ***Via ETON Exit*:** 1st drop-off point – Laguna Blvd. Guard House (across Paseo PHOENIX Gasoline Station) 2nd drop-off point – Milagros Del Rosario (MRR) Building - East Canopy
DLSU Laguna Campus - Milagros Del Rosario (MRR) Building - East Canopy	1st drop-off point – Petron Gasoline Station along Gil Puyat Avenue • ***Via Estrada*:** 2nd drop-off point – Gate 4: Gokongwei Gate 3rd drop-off point – Gate 2: North Gate (HSSH) 4th drop-off point – Gate 1: South Gate (LS Building Entrance) • ***Via Buendia/LRT*:** 2nd drop-off point – College of St. Benilde (CSB) along Taft Avenue 3rd drop-off point – Gate 4: Gokongwei Gate 4th drop-off point – Gate 2: North Gate (HSSH) 5th drop-off point – Gate 1: South Gate (LS Building Entrance)

NOTE: Passengers are NOT ALLOWED to board from any of the drop-off points.

The use of shuttle service shall ONLY be intended for passengers **with Inter-campus and/or official business transactions** at Manila and Laguna campuses. All DLSU employees and students (undergraduate and graduate school) may ride Arrows Express Line 1, following the priority groups for seat reservations below:

Priority Groups	
1	Faculty and ASF with Inter-campus assignments
2	Students with Inter-campus enrolled subjects or enrolled in thesis using Inter-campus facilities
3	Researchers

4	School Administrators (Academic Coordinators level and up for Faculty and ASF, and Director level and up for APSP)
5	University Fellows
6	Employees and Students with official business

A passenger will have to first fill up an embarkation card with the following information:

- Current Date
- Trip number
- Embarkation Point
- Name
- ID Number
- Priority Group Number
- Drop-off point (depending on the route)

The shuttle can accommodate up to 13 passengers.

1	2	3
4	5	6
7	8	9
10	11	12
13	Driver	

Fig. 1 Regular seating

During worst case scenarios (lots of passengers for a specific time), they accommodate up to 16 passengers.

1	2	3	4
5	6	7	8
9	10	11	12
	13	14	15
16		Driver	

Fig. 2 Worst case scenario seating

The priority levels are strictly adhered to during each trip. Meaning those with higher priority will be the first in line when loading onto the shuttle.

You are tasked with creating a program to keep track of the trips of the shuttles.

The system will always check the priority level of each new passenger entry and adjust the seating accordingly. This means that it is possible for those with a lower priority level to be removed from a specific trip and moved to the next.

The system will have the following features depending on the user:

Passenger Features
● Embarkation encoding ○ It will ask the user for the **Trip Number**. ○ This is where the user will encode the embarkation information that was mentioned above. (Trip Number, ○ The system will automatically assign the passenger to the selected trip if that trip is not yet full. In the event that the trip is full and a high priority passenger wants to board, the first lowest priority passenger listed will be removed and moved to the next trip. The removed passenger will be notified(display a message on screen). **NOTE: Seat assignment will start at seat 1. Refer to the figures above.**

Feature for both user types
● Notify/Inform the user that a specific trip is full ○ When the user picks a specific trip, if the trip is full, he/she will be notified via display message.

Arrows Express Personnel Features
● Viewing the number of passengers on a certain trip ○ It will ask the user for the **Trip Number**. ○ This will display the current count for a specific trip. ○ Display will be as seen above. All taken seats will be marked with a “X” while open/available seats will be marked with an “O”. ● View the count of passengers of a specific drop-off point ○ It will ask the user for the **Trip Number**. ○ This will display a list of the stops that the passengers picked and show how many will be getting off at those stops. ● View passenger information ○ This feature will ask for the **Trip Number**. ○ It will then display a list of the names, ID Numbers and priority numbers of the passengers of the specific trip sorted by priority number(highest to lowest). ● Load passenger information ○ This will be an alternative to manually inputting passenger information. ○ It will read the trip number as well as the passenger info from a text file. ● Search passenger ○ This feature will allow the user to search for passengers by their last name. ● Automatically save current information to a text file when the program is terminated. ● View any recent trip files via a file load feature. ○ Using this feature will display all information of the loaded trip file.

The information should be saved in Trip-dd-mm-yyyy.txt with the following format:

```
<trip number><new line>
<embarkation point><new line>
<passenger name><new line>
<id number><new line>
<priority number><new line>
<date of trip><new line>
<drop-off point><new line>
.
.
<trip number><new line>
<embarkation point><new line>
<passenger name><new line>
<id number><new line>
<priority number><new line>
<drop-off point><new line>
<new line>
.
.
.
<eof>
```

Where:

dd – current day
mm – current month
yyyy – current year

Saving is done on a daily basis, meaning each day is 1 file.

Submission & Demo

- **Final MP Deadline: June 17, 2022 (F), 1200noon via Canvas**. After the indicated time, the submission page will be locked and thus considered no submission (equivalent to 0 in the MP grade).

  Requirements: Complete Program
  - Make sure that your implementation has considerable and proper use of arrays, strings, structures, files, and user-defined functions, as appropriate, even if it is not strictly indicated.
  - It is expected that each feature is supported by at least one function that you implemented. Some of the functions may be reused (meaning you can just call functions you already implemented [in support] for other features. There can be more than one function to perform tasks in a required feature.
  - Debugging and testing was performed exhaustively. The program submitted has
    a. NO syntax errors
    b. NO warnings - make sure to activate -Wall (show all warnings compiler option) and that C99 standard is used in the codes
    c. NO logical errors -- based on the test cases that the program was subjected to

Important Notes:

1. Use **gcc -Wall** to compile your C program. Make sure you **test** your program completely (compiling & running).
2. Do not use brute force. Use **appropriate conditional** statements **properly**. Use, **wherever appropriate**, **appropriate loops** & **functions properly**.
3. You **may** use topics outside the scope of CCPROG2 but this will be **self-study**. Goto *label*, exit(), break (except in switch), continue, global variables, calling main() are **not allowed**.
4. Include **internal documentation** (comments) in your program.
5. The following is a checklist of the deliverables:

Checklist:
- Upload via AnimoSpace submission:
  - source code*
  - test script**
  - sample text file exported from your program
- email the softcopies of all requirements as attachments to **YOUR** own email address on or before the deadline

Legend:

***** Source code exhibit readability with supporting inline documentation (not just comments before the start of every function) and follows a coding style that is similar to those seen in the course notes and in the class discussions. The first page of the source code should have the following declaration (in comment) [replace the pronouns as necessary if you are working with a partner]:

/***
This is to certify that this project is my own work, based on my personal efforts in studying and applying the concepts learned. I have constructed the functions and their respective algorithms and corresponding code by myself. The program was run, tested, and debugged by my own efforts. I further certify that I have not copied in part or whole or otherwise plagiarized the work of other students and/or persons.

<your full name>, DLSU ID# <number>
***/

Example coding convention and comments before the function would look like this:

```
/* funcA returns the number of capital letters that are changed to small letters
   @param strWord - string containing only 1 word
   @param pCount - the address where the number of modifications from capital to small are
                   placed
   @return 1 if there is at least 1 modification and returns 0 if no modifications
   Pre-condition: strWord only contains letters in the alphabet
*/
int     //function return type is in a separate line from the
funcA(char strWord[20] ,    //preferred to have 1 param per line
      int * pCount)            //use of prefix for variable identifiers
{  //open brace is at the beginning of the new line, aligned with the matching close brace
   int    ctr;        /* declaration of all variables before the start of any statements –
                         not inserted in the middle or in loop- to promote readability */

   *pCount = 0;
   for (ctr = 0; ctr < strlen(strWord); ctr++)  /*use of post increment, instead of pre-
                                                  increment */
   {  //open brace is at the new line, not at the end
      if (strWord[ctr] >= 'A' && strWord[ctr] <= 'Z')
      {    strWord[ctr] = strWord[ctr] + 32;
           (*pCount)++;
      }
      printf("%c", strWord[ctr]);
   }

   if (*pCount > 0)
      return 1;
   return 0;
}
```

******Test Script should be in a table format. There should be at least 3 categories (as indicated in the description) of test cases **per function**. There is no need to test functions which are only for screen design (i.e., no computations/processing; just printf).

Sample is shown below.

Function	**#**	**Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
sortIncreasing	1	Integers in array are in increasing order already	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	P
	2	Integers in array are in decreasing order	aData contains: 53 37 33 32 15 10 8 7 3 1	aData contains: 1 3 7 8 10 15 32 33 37 53	aData contains: 1 3 7 8 10 15 32 33 37 53	P
	3	Integers in array are combination of positive and negative numbers and in no particular sequence	aData contains: 57 30 -4 6 -5 -33 -96 0 82 -1	aData contains: -96 -33 -5 -4 -1 0 6 30 57 82	aData contains: -96 -33 -5 -4 -1 0 6 30 57 82	P

Test descriptions are supposed to be unique and should indicate classes/groups of test cases on what is being tested. Given the sample code in page 6, the following are four distinct classes of tests:

i.) testing with strWord containing all capital letters (or rephrased as "testing for at least 1 modification")
ii.) testing with strWord containing all small letters (or rephrased as "testing for no modification")
iii.) testing with strWord containing a mix of capital and small letters
iv.) testing when strWord is empty (contains empty string only)

The following test descriptions are **incorrectly formed**:

- Too specific: testing with strWord containing "HeLlo"
- Too general: testing if function can generate correct count OR testing if function correctly updates the strWord
- Not necessary -- since already defined in pre-condition: testing with strWord containing special symbols and numeric characters

6. Upload the softcopies via Submit Assignment in Canvas. You can submit multiple times prior to the deadline. However, only the last submission will be checked. Send also to your **mylasalle account** a **copy** of all deliverables.

7. Use **<surnameFirstInit>.c** & **<surnameFirstInit >.pdf** as your filenames for the source code and test script, respectively. You are allowed to create your own modules (.h) if you wish, in which case just make sure that filenames are descriptive.

8. During the MP **demo**, the students are expected to appear on time, to answer questions, in relation with the output and to the implementation (source code), and/or to revise the program based on a given demo problem. Failure to meet these requirements could result to a grade of 0 for the project.

9. It should be noted that during the MP demo, it is expected that the program can be compiled successfully and will run. If the program does not run, the grade for the project is automatically 0. However, a running program with complete features may not necessarily get full credit, as implementation (i.e., code) and other submissions (e.g., non-violation of restrictions evident in code, test script, and internal documentation) will still be checked.

10. The MP should be an HONEST intellectual product of the student/s. For this project, you are allowed to do this individually or to be in a group of 2 members only. Should you decide to work in a group, the following mechanics apply:

Individual Solution: Even if it is a group project, each student is still required to create his/her INITIAL solution to the MP individually without discussing it with any other students. This will help ensure that each student went through the process of reading, understanding, solving the problem and testing the solution.

Group Solution: Once both students are done with their solution: they discuss and compare their respective solutions (ONLY within the group) -- note that learning with a peer is the objective here -- to see a possibly different or better way of solving a problem. They then come up with their group's final solution -- which may be the solution of one of the students, or an improvement over both solutions. Only the group's final solution, with internal documentation (part of comment) indicating whose code was used or was it an improved version of both solutions) will be submitted as part of the final deliverables. It is the group solution that will be checked/assessed/graded. Thus, only 1 final set of deliverables should be uploaded by one of the members in the Canvas submission page. [Prior to submission, make sure to indicate the members in the group by JOINing the same group number.]

Individual Demo Problem: As each is expected to have solved the MP requirements individually prior to coming up with the final submission, both members should know all parts of the final code to allow each to INDIVIDUALLY complete the demo problem within a limited amount of time (to be announced nearer the demo schedule). This demo problem is given only on the day/time of the demo, and may be different per member of the group. Both students should be present/online during the demo, not just to present their individual demo problem solution, but also to answer questions pertaining to their group submission.

Grading: the MP grade will be the same for both students -- UNLESS there is a compelling and glaring reason as to why one student should get a different grade from the other -- for example, one student cannot answer questions properly OR do not know where or how to modify the code to solve the demo problem (in which case deductions may be applied or a 0 grade may be given -- to be determined on a case-to-case basis).

11. Any form of **cheating** (**asking other people not in the same group for help**, **submitting as your [own group's] work part of other's work**, **sharing your [individual or group's] algorithm and/or code to other students not in the same group, etc.**) can be punishable by a grade of **0.0** for the **course** & a **discipline case**.

12. **Not having an MP demo** will merit a **zero(0) for your MP grade**.

13. **Bonus points:** You can earn up to **at most 10 points** bonus for additional features that your program may have. Any additional features may be added but should not conflict with whatever instruction was given in the project specifications. Bonus points are given upon the discretion of the teacher, based on the difficulty and applicability of the feature to the program. Note that bonus points can only be credited if all the basic requirements are fully met (i.e., complete and no bugs).

Any requirement not fully implemented or instruction not followed will merit deductions.

----------There is only 1 deadline for this project: June 17, 12:00 noon----------