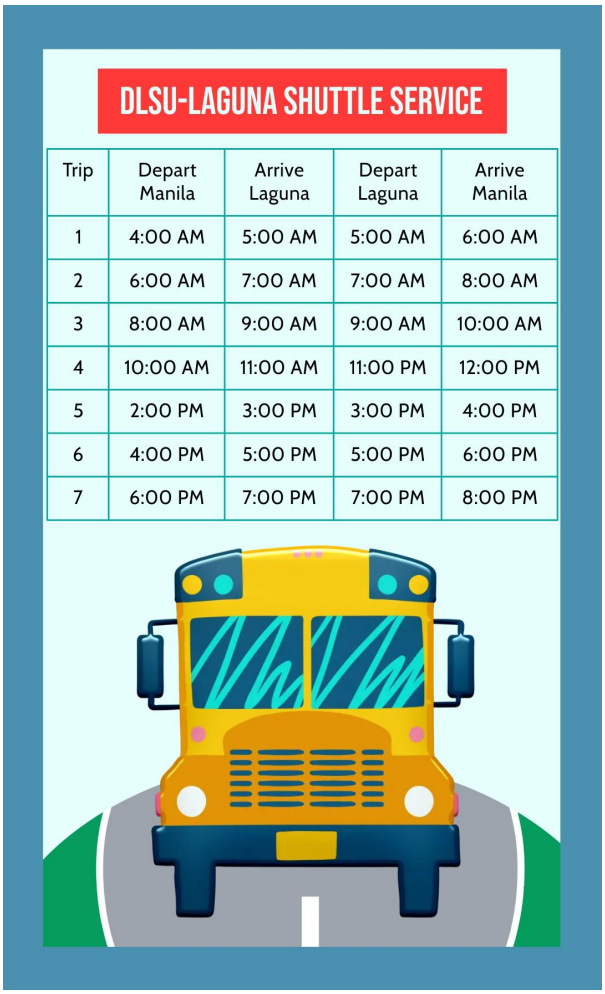
De La Salle University
College of Computer Studies
Software Technology Department



Trip	Depart Manila	Arrive Laguna	Depart Laguna	Arrive Manila
1	4:00 AM	5:00 AM	5:00 AM	6:00 AM
2	6:00 AM	7:00 AM	7:00 AM	8:00 AM
3	8:00 AM	9:00 AM	9:00 AM	10:00 AM
4	10:00 AM	11:00 AM	11:00 PM	12:00 PM
5	2:00 PM	3:00 PM	3:00 PM	4:00 PM
6	4:00 PM	5:00 PM	5:00 PM	6:00 PM
7	6:00 PM	7:00 PM	7:00 PM	8:00 PM

Shuttle Service Ticket Booking

CCPROG1 MACHINE PROJECT
(Deadline of submission on/before 0800 of November 28)

With our continuously improving modern world, digitalization is apparent. Digitalization refers to the use of digital technologies which is used in order to improve operations in businesses and also to create new value for their customers. But before this usable technologies come digitization - the process of converting physical information into digital formats.

For this problem, you are to digitize buying of bus tickets from a ticket agent. Your problem analysis and logic formulation are expected to be practiced in developing this application.
**Note that this does not follow the idea of actual DLSU shuttle.*

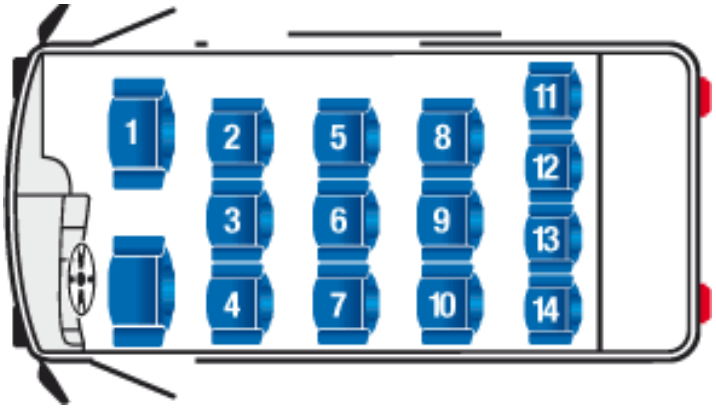
MENU

The application should be menu-driven to be used by a single human teller where riders queue in. There will be five menus that are required for this project.

- Book a ticket
- Cancel a booking
- Display Schedule
- Update Departure Time
- Close application

Book a ticket

This is where the ticket agent can 1) see some information and 2) book an available seat in the current bus schedule *(please refer to the image above)*.



Displayed information must include the **departure time**, **route** (if from Manila to Laguna or vice versa), and **number of available seats**. By default, your bus only contains 14 seats. If all seats are already taken, there should be a display saying so.

To book a ticket, the agent specifies the seat number and student's id number. If already taken, then it cannot be booked anymore and necessary steps should be done.

Cancel a booking
After successful booking, the agent can also cancel a reservation.
On this page, the booked seats with the id number should also be displayed. If the chosen seat is not yet booked, then it cannot be canceled. If it is booked, then the chosen seat will again be available in the 'Book a ticket' page.

Display Schedule
This displays all the 14 bus schedules *(as per the image in the previous page)*. There should be a clear distinction which bus schedule was already chosen (past), the current one, and the possible next departure times.

Update Departure Time
This is the page where the agent can manually change the current departure time.
The new input time can only be future time, not a previous one. For example, times 4AM, 5AM, and 8AM are already chosen therefore the agent can only choose from times 8AM above, 6AM and 7AM cannot be accepted.
TAKE NOTE: *Once departure time is changed, bookings on the previous schedule will be deleted. Your program should also automatically detect the route (Manila to Laguna & Laguna to Manila) based on the departure time.*
Hint: *As per the first image in the previous page, there are 7 schedules for each route. Notice that the Manila to Laguna schedules' departure time is always an even number while Laguna to Manila is an odd number.*

Close application
By choosing to close the application, there should be a display message first to inform the agent of deletion of records (bookings) before actually ending the execution.

YOUR TASK

Create this simple ticket booking application in C programming language considering the above given specifications and the following statements:

1. A welcome page displaying the 5 menus is shown.
2. On the first run, the departure time is 4AM by default.
3. All seats are available during the program start and when the departure time is changed.
4. An information/error message is displayed in every action such as *"Booking successful"*, *"Booking cannot be made as seat is not available"*, *"Not possible departure time",* etc.
5. A bus layout with fourteen seats is displayed in both **'Book a Ticket'** and **'Cancel a Ticket'** options. This layout can use any symbols but make sure that there is a visual representation of which seat is available and not. Feel free to use ASCII art but double check if it is available to all operating systems.
6. **'Close application'** option should display a closing screen and/or message plus how many tickets were booked during the run before terminating the whole program.
7. Everything in this application should use conditional and looping statements. User-defined functions should also be declared and called. No brute force implementation is allowed in any part of the application.

How to Approach the Machine Project

Step 1: Problem analysis and algorithm formulation

Read the MP Specifications again! Identify clearly what are the required information from the user, what kind of processes are needed, and what will be the output(s) of your program. Clarify with your professor any issues that you might have regarding the machine project.

When you have all the necessary information, identify the necessary functions that you will need to modularize the project. Identify the required data of these functions and what kind of data they will return to the caller. Write your algorithm for each of these modules/functions as well as the algorithm for your main program.

Step 2: Implementation

In this step, you are to translate your algorithm into proper C statements. While implementing, you are to perform the other phases of program planning and design (discussed in the other steps below) together with this step.

Follow the Linux Kernel coding standard.

You may choose to type your program in a text editor or an IDE (i.e. Dev-C IDE) at this point. Note that you are expected to use statements taught in class. You can explore other libraries and functions in C as long as you can clearly explain how these work. You may also use arrays, should these be applicable and you are able to properly justify and explain your implementation using these. For topics not covered, it is left to the student to read ahead, research, and explore by himself.

Note though that you are **NOT ALLOWED** to do the following:

- to declare and use global variables (i.e., variables declared outside any function),
- to use `goto` statements (i.e., to jump from code segments to code segments),
- to use the `break` statement to exit a block other than `switch` blocks,
- to use the `return` statement or `exit` statement to **prematurely** terminate a loop or function or program,
- to use the `exit` statement to **prematurely** terminate a loop or to terminate the function or program, and
- to call the `main()` function to repeat the process instead of using loops.

It is best that you perform your coding “incrementally.” This means:

- dividing the program specification into subproblems, and solving each problem separately according to your algorithm;
- coding the solutions to the subproblems one at a time. Once you’re done coding the solution for one subproblem, apply testing and debugging.

Documentation

While coding, you have to include internal documentation in your programs. You are expected to have the following:

- File comments or Introductory comments
- Function comments
- In-line comments

Introductory comments are found at the very beginning of your program before the preprocessor directives. Follow the format shown below. Note that items in between **< >** should be replaced with the proper information. Items in between ***[ ]*** are optional, indicate if applicable.

```
/*
     Description: <Describe what this program does briefly>
     Programmed by: <your name here> <section>
```

```
     Last modified: <date when last revision was made>
     Version: <version number>
     [Acknowledgements: <list of sites or borrowed libraries and sources>]
*/

<Preprocessor directives>

<function implementation>

int main()
{
     return 0;
}
```

Function comments precede the function header. These are used to describe what the function does and the intentions of each parameter and what is being returned, if any. If applicable, include pre-conditions as well. Pre-conditions refer to the assumed state of the parameters. Follow the format below when writing function comments:

```
/*   <Description of function>
     Precondition: <precondition / assumption>
     @param <name> <purpose>
     @return <description of returned result>
*/
<return type>
<function name> (<parameter list>)
     :
```

Example:

```
/* This function computes for the area of a triangle
   Precondition: base and height are non-negative values
   @param base is the base measurement of the triangle in cm
   @param height is the height measurement of the triangle in cm
   @return the resulting area of the triangle
*/
float
getAreaTri (float base,
            float height)
{
    ...
}
```

In-Line comments are other comments in major parts of the code. These are expected to explain the purpose or algorithm of groups of related code, esp. for long functions.

Step 3: Testing and Debugging

SUBMIT THE LIST OF TEST CASES YOU HAVE USED.

For each feature of your program, you have to fully test it before moving to the next feature. Sample questions that you should ask yourself are:

1) What should be displayed on the screen after a user input?
2) What would happen if inputs are incorrect? (e.g., values not within the range)
3) Is my program displaying the correct output?
4) Is my program following the correct sequence of events (correct program flow)?
5) Is my program terminating (ending/exiting) correctly? Does it exit when I press the command to quit? Does it exit when the program's goal has been met? Is there an infinite loop?
6) and others...

IMPORTANT POINTS TO REMEMBER:

1. You are required to implement the project using the C language (**C99** and NOT C++). Make sure you know how to compile and run in both the IDE (DEV-C++) and the command prompt via

   `gcc -Wall -std=c99 <yourMP.c> -o <yourExe.exe>`

2. The implementation will require you to:
   - Create and Use Functions

     Note: Non-use of self-defined functions will merit a grade of **0** for the **machine project**.
   - Appropriately use conditional statements, loops and other constructs discussed in class (Do not use brute force solution. You are not allowed to use `goto` label statements, `exit` statements. You are required to pass parameters to functions and not allowed to declare global or static variables.)
   - Consistently employ coding conventions
   - Include internal documentation (i.e., comments)
3. Deadline for the project is the **7:59AM of November 28, 2022 (Tuesday)** via submission through **AnimoSpace**. After this time, the submission facility is locked and thus no MP will be accepted anymore and this will result to a **0** for your **machine project**.
4. The following are the deliverables:

   Checklist:
   - Upload in **AnimoSpace** by clicking **Submit Assignment** on Machine Project and adding the following files:
     - source code*
     - test script**
   - email the softcopies of everything as attachments to YOUR own email address on or before the deadline

   Legend:
   * Source Code also includes the internal documentation. The **first few lines of the source code** should have the following declaration (in comment) **BEFORE** the introductory comment:

```
/*****************************************************************
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>
*****************************************************************/
```

** Test Script should be in a table format, with header as shown below. There should be at least 3 distinct test classes (as indicated in the description) **per function**. There is no need to test functions which are only for screen design.

Function Name	#	Test Description	Sample Input *(either from the user or passed to the function)*	Expected Result	Actual Result	P/F
`getAreaTri()`	1	base and height measurements are less than 1	base = 0.25 height = 0.75	...	...	...
	2					
	3					

5. **MP Demo:** You will demonstrate your project on a specified schedule during the last weeks of classes. Being unable to show up on time during the demo or being unable to answer convincingly the questions during the demo will merit a grade of **0** for your **machine project**. The project is initially evaluated via black box testing (i.e., based on output of running program). Thus, if the program does not compile successfully using gcc and execute in the command prompt, a grade of **0** for the project will be incurred. However, a fully working project does not ensure a perfect grade, as the implementation (i.e., correctness and compliance in code) is still checked.
6. Any requirement not fully implemented and instruction not followed will merit deductions.
7. This is an **individual project**. Working in collaboration, asking other people's help, borrowing or copying other people's work or from books or online sources (either in full or in part) are considered as cheating. Cheating is punishable by a grade of **0.0** for **CCPROG1** course. Aside from which, a cheating case may be filed with the Discipline Office.
8. **Bonus points:** You can earn up to **at most 10 points** bonus for additional features that your program may have. Sample features and implementation that may allow you to gain bonus points include: **option to change bus type with different number of seats for any schedule**. Some of the indicated additional features may require self-study. Also, any additional feature not stated here 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).