

Machine Project Specs

Ups and Downs

Outline

- Introduction
- Board
 - Regular Tile
 - UP/DOWN Tile
- Setting Up the Game

Introduction

Introduction

- For this project, you are to create a game called Ups and Downs. This is a turn-based two-player board game. Wherein the goal of each player is to reach the end point of the board. The first one to reach the end point wins the game.


Board

Board – Regular Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	9


- The board in this game is made up to 10x10 cells, where each cell is given a numerical label as shown above. The label of each cell indicate how players will traverse the board.

Board – Regular Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	


- For example, if player 1 is in tile 9, the next tile he'll traverse is tile 10.

Board – Regular Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	
0	1	2	3	4	5	6	7	8	9


- For example, if player 1 is in tile 9, the next tile he'll traverse is tile 10.

Board – Regular Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12		10
0	1	2	3	4	5	6	7	8	9


- For example, if player 1 is in tile 9, the next tile he'll traverse is tile 10. Then 11,

Board – Regular Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13		11	10
0	1	2	3	4	5	6	7	8	9


- For example, if player 1 is in tile 9, the next tile he'll traverse is tile 10. Then tile 11, tile 12

Board – Regular Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14		12	11	10
0	1	2	3	4	5	6	7	8	9


- For example, if player 1 is in tile 9, the next tile he'll traverse is tile 10. Then tile 11, tile 12, and so on until he reaches tile 99.

Board – Up/Down Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14		12	11	10
0	1	2	3	4	5	6	7	8	9


- Aside from the regular tiles, there are also up and down tiles that teleports a player either up or down the board.

Board – Up Tile (Scenario 1)

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14		12	11	10
0	1	2	3	4	5	6	7	8	9


- For example, if tile 14 is an UP tile connected to tile 50, and player 1 steps on tile 14, he'll automatically be transported to tile 50.

Board – Up Tile (Scenario 1)

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15		13	12	11	10
0	1	2	3	4	5	6	7	8	9


- For example, if tile 14 is an UP tile connected to tile 50, and player 1 steps on tile 14, he'll automatically be transported to tile 50.

Board – Up Tile (Scenario 1)

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	9


- For example, if tile 14 is an UP tile connected to tile 50, and player 1 steps on tile 14, he'll automatically be transported to tile 50.

Board – Up Tile (Scenario 2)

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	9


- On the other hand, if player 1 is on tile 49 and moves to tile 50, he won't be transported back to tile 14 since tile 14 is a UP tile.

Board – Up Tile (Scenario 2)

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	9


- On the other hand, if player 1 is on tile 49 and moves to tile 50, he won't be transported back to tile 14 since tile 14 is a UP tile.

Board – Down Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	9


- DOWN tiles are like UP tiles except, it teleports a player down to a lower-valued tile.

Board – Down Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56		54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	9


- For example, if tile 56 is DOWN tile connected to tile 8 then if player 1 happens to step on tile 56, he will be transported back to tile 8.

Board – Down Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57		55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	9

- For example, if tile 56 is DOWN tile connected to tile 8 then if player 1 happens to step on tile 56, he will be transported back to tile 8.

Board – Down Tile

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7		9

- For example, if tile 56 is DOWN tile connected to tile 8 then if player 1 happens to step on tile 56, he will be automatically transported back to tile 8.

Board – UP/Down Representation

99	98	97	96	95	94	93	92	91	90
80	81	82	83	84	85	86	87	88	89
79	78	77	76	75	74	73	72	71	70
60	61	62	63	64	65	66	67	68	69
59	58	57	56	55	54	53	52	51	50
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	31	30
20	21	22	23	24	25	26	27	28	29
19	18	17	16	15	14	13	12	11	10
0	1	2	3	4	5	6	7	8	9

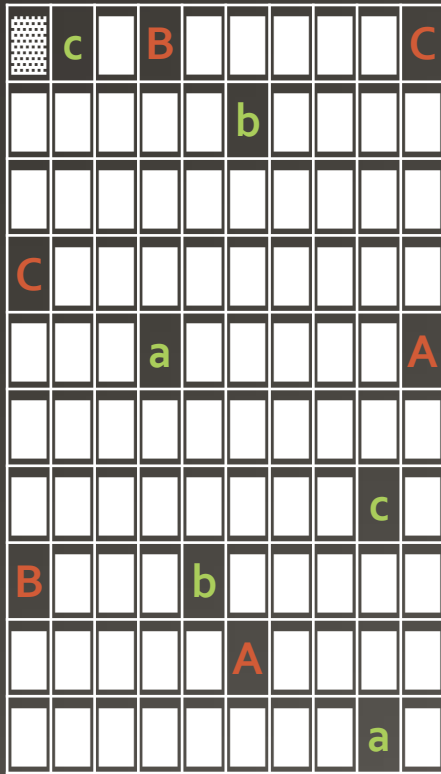
- Here in the project specifications, we represent the connection of tiles in terms of colors. However, in your program, you may use ASCII symbols or letters to represent them.
- You may refer to this [website\(click me\)](#) for the list of available ASCII symbols.



Board – UP/Down Representation

99	c	97	B	95	94	93	92	91	C
80	81	82	83	84	b	86	87	88	89
79	78	77	76	75	74	73	72	71	70
C	61	62	63	64	65	66	67	68	69
59	58	57	a	55	54	53	52	51	A
40	41	42	43	44	45	46	47	48	49
39	38	37	36	35	34	33	32	c	30
B	21	22	23	b	25	26	27	28	29
19	18	17	16	15	A	13	12	11	10
0	1	2	3	4	5	6	7	a	9

- For example: small letters represent DOWN tiles, capital letters represent UP tile, and tiles of the same letter are connected tiles.
- NOTE: In your project, you are **NOT** required to use colors to represent these types. The colors are there so you can easily see the UP/DOWN tiles in the board.

Board – Representation



- You may also use ASCII symbols to represent regular tiles in the board. Just make sure that the players are traversing the board accordingly.
- In the given example:
 -  <- Goal Tile
 -  <- Regular Tile


Board – Hard Requirements

- Your board should have at least 6 Pairs of UP-DOWN (3 UP to DOWN and 3 DOWN to UP) Tiles scattered. They must be randomly generated by your program and ensure that they are placed correctly. The following should be checked by your program:
 - UP Tile should be from lower to higher and not the other way around
 - DOWN Tile should be from higher to lower and not the other way around
 - Last tile on the board should be a DOWN tile
 - A row can only have 1 UP TILE and 1 DOWN TILE, at most
 - The destination of UP Tile and DOWN Tile could be placed anywhere as long as the UP tile boosts the player and the DOWN tile drags the player back to a previous position
- Be consistent with your symbols (e.g., Capital letters indicate UP tiles, small letters indicate DOWN tiles)

Setting Up the Game

Setting up the Game

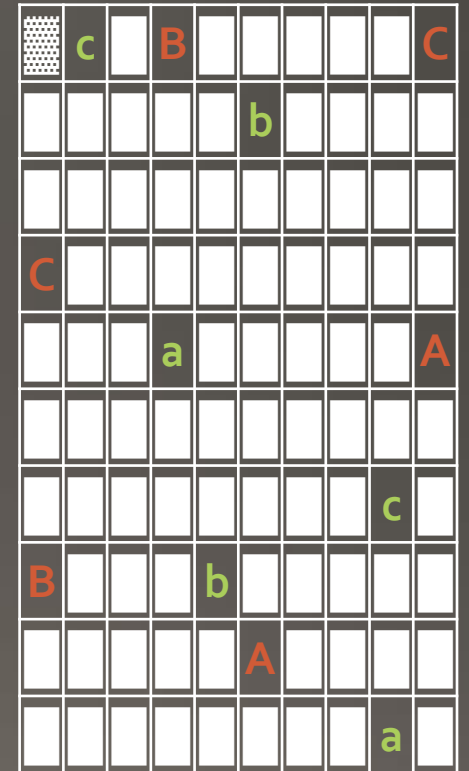
- There are only 2 players in the game. At the start of the game, each player starts at tile 0.
- You may choose any ASCII Symbol to represent your player 1 and 2.
 - E.g., Capital letters represent UP Tiles and small letters represent DOWN tiles

	c		B						C
					b				
C									
			a						A
								c	
B				b					
					A				
								a	

Playing the Game

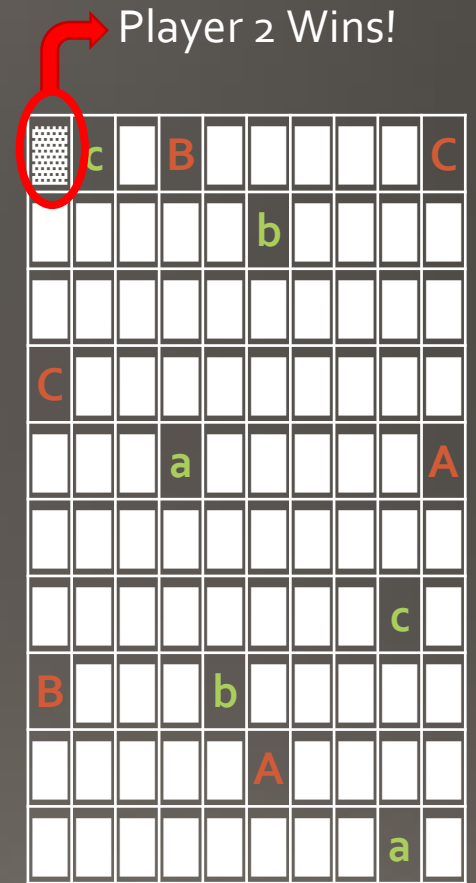
Playing the Game

- UP and DOWN is a turn-based game wherein the goal of each player is to be the first to reach the goal tile.
- At the start of their turn, the player rolls the die to determine how many steps he's going to take.
 - For example, if player 1 rolls 3, then he takes 3 steps forward on the board.
- The die in the game is special, contrary to the real die, the die in this game has a 0 side. This means that the player does not take any steps forward.
- **Note: You will have to learn how to use rand() or srand() function to generate a random number to simulate the rolling of the die**



Playing the Game

- In case the 2 players are on the same tile, use the character '@' to represent that the players are on the same tile.
- Once a player reaches the goal tile, that player wins that game



Playing the Game

- Once the player steps out of the tile, make sure that the representation of the tile is not lost.

	c		B						C
					B				
C									
			a						A
								c	
B				b					
					A				
2			1					a	



	c		B						C
					B				
C									
			a						A
								c	
B				b					
					A				
2					1			a	



CCPDOC, Term 1 AY 2023 - 2024

WRONG!, unfilled



	c		B						C
					B				
C									
			a						A
								c	
B				b					
					A				
2			1					a	



	c		B						C
					b				
C									
			a						A
								c	
B				b					
					A				
2					1			a	



CORRECT!

Special Rule

- If a player rolls 1 consecutively for 2 turns, he'll have another turn to roll the die and the game will still proceed accordingly.

Bonus Features

1. Players may start a new game once a game has ended
2. Before the start of the game, each player may place a DOWN tile and its link anywhere on the board and the opponent player (only) who steps on it is transported back in the board. Player may not place the link of their DOWN tile in first row of the board.

Documentation: Code

Documentation: Code

- You are expected to have the following internal documentation in your code:
 - Introductory Comments
 - Function Comments
 - In-line Comments
- They are typically placed in a multiline comment (`/* */`)

Documentation: Introductory Comments

- Introductory Comment – found at the beginning of your program before the preprocessor directives. Typically, it contains the following fields:
 - Description – briefly describes what the program does
 - Author – name of the programmer
 - (Optional) Acknowledgements - List of sites or borrowed libraries and sources
-

Example:

```
/*  
    Description: This program computes for the radius of a circle  
    Author: Juan dela Cruz - S19  
    Acknowledgements:  
*/  
#include <stdio.h>  
int main() {  
    return 0;  
}
```

Documentation: Function Comments

- Function comments – Precedes the function header. Describes what the function does. Typically, it contains the following fields:
 - (Optional) Precondition – assumed state of the parameters
 - @param – name of the parameter and purpose
 - @return – description of the value returned

Example:

```
/*  
    Precondition: parameters are non-negative  
    @param: height - height of the rectangle in  
cm  
    @param: width - width of the rectangle in cm  
    @return: computed area of the rectangle  
*/  
float getRectangleArea(float height, float width) {  
    return height * width;  
}
```

Documentation: In-line Comments

- In-line comments – Additional comments in the code explaining the purpose or algorithm

Example:

```
float getRectangleArea(float height, float width) {  
    return height * width; // area of rectangle = base * height  
}
```

Documentation: Test Script

Test Script

- Create a list of test cases you used to test your machine project. For this project we are going to use the following testing techniques:
 - Unit Testing – done for each function in the program
 - System Testing – done for each feature in the program
- For this project, you are only required to document black-box testing results, which means you give an input to the function or system and see if the expected output is satisfied. Nonetheless, you are still responsible for ensuring that that codes inside your program is working according to specs.
- You may refer to the following website for additional details:
https://www.softwaretestinghelp.com/types-of-software-testing/#1_Unit_Testing

Test Script – Unit Testing

- Identify the boundary values, and representative values before and after the boundary test case

Example Documentation for 1 Function:

These test cases failed because the expected output is not the same as the actual output. This means there's a bug in your program that needs to be addressed

Function Name: int getAgeGroupType (int ageInMonths)					
Function Description: Returns the age group type of person base on age input					
#	Test Description	Sample Input either from the user or passed to the function	Expected Results	Actual Results	Pass / Fail (P/F)
1	Test Invalid Age Input (Age < 0)	-1	-1	-1	P
2	Test Toddler Lower Boundary Age (Age = 0)	0	0	-1	F
3	Test Toddler In Between Age (0 < Age < 12)	1	0	0	P
4	Test Toddler Upper Boundary Age (Age = 12)	12	0	0	P
5	Test Children Lower Boundary Age (Age = 13)	13	1	0	F
...

Test Script – System Testing

- System testing is like unit testing except it is based on the features in the program rather on a per function. Clearly, this implies testing the interaction of multiple function features often involve multiple functions in it.
- For this project, are you may force the die to take a certain value rather than having it randomly generated to test a feature

Test Script – System Testing

- Example Documentation for 1 Feature:

Feature name: DOWN Tile					
#	Test Description	Sample Input either from the user or passed to the function	Expected Results	Actual Results	Pass / Fail (P/F)
1	Test Player 1 steps on a DOWN Tile	15 (* Assuming that a DOWN tile is tile 15)	Player 1 is moved back to Tile 5 (*Assuming that DOWN Tile in 15 is linked to tile 5)	Player 1 is moved back to Tile 5	P
2	Test Player 1 steps on a tile a link of a DOWN Tile	5	Player 1 remains in Tile 5	Player 1 is moved up to Tile 15	F
3	Test Player 2 steps on a DOWN Tile placed by Player 1	30	Player 2 is moved back to Tile 10	Player 2 is moved back to Tile 10	P
...

Implementation Restrictions

Implementation Restrictions

- You the following are **not allowed** in your machine project:
 - Use **Arrays** and **Strings** for this project
 - 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 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.
- Non-compliance to these restrictions may merit a grade of 0 for the machine project component of CCPROG1.

Reminders

Reminders

1. Your program should compile using the command in CMD as show below. Otherwise, you will automatically get a grade of 0 for your Machine Project.

```
gcc -Wall -std=c99 <Your Code File Name>.c -o <Your Exe File>.exe
```

2. Make sure that you are using C Language applicable in C99 standard and not C++ or any other variations of C.

3. Your MP code must have at least 5 functions in it.

4. Ensure that you take not the implementation restrictions found in the previous section. You are required to pass/return parameters in functions. You are not allowed to use global and static variables.

5. Follow the coding convention (format)

6. Document your code (comments) and test script

Reminders

7. MP Demo: You will demo your project in the last weeks of the classes. Unable to show up on time during your demo schedule or failure to provide convincing answers to questions during demo will merit a group of 0 for your machine project. Your MP demo usually consists of 2 parts – black box (feature demo-based) and white box (code-based). This means that a fully working project does not ensure a perfect grade as the way your project was implemented is also examined (e.g., coding convention, presence of functions, etc.).

8. Any requirement not implemented, or partially implemented and unfollowed instructions will merit deductions.

9. **THIS IS AN INDIVIDUAL PROJECT.** Working in collaboration, asking other people's help, using AI such as chatbots, borrowing or copying other people's work from books or online resources either in full or partially are considered cheating. Thus, this is punishable by a grade of 0.0 for CCPROG1. Aside from which, a cheating case may be filed in the Discipline Office.

Reminders

10. You may earn up to 10 points bonus points (max) for this project for additional features you may have implemented. The number of points to be awarded will depend on the complexity of the added feature. This will be decided by your instructor during MP demo. Additional features will be considered only if they are not in conflict with the primary features of the program as written in this project specs.

Note that bonus points will only be considered if all the basic requirements are fully met.

Reminders

11. Ensure that you have uploaded and submitted the following in AnimoSpace:

- Source Code in C
- Test Script in PDF
- Declaration of Original Work in PDF

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 all 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>

12. Email items in Step 11 to your own DLSU email on or before the Machine Project Deadline as backup

13. Deadline for this project is on November 28, 2023 (08:00 AM)