A.Y. 2023-2024 | Second Semester

CIS 1201 (Programming 2)

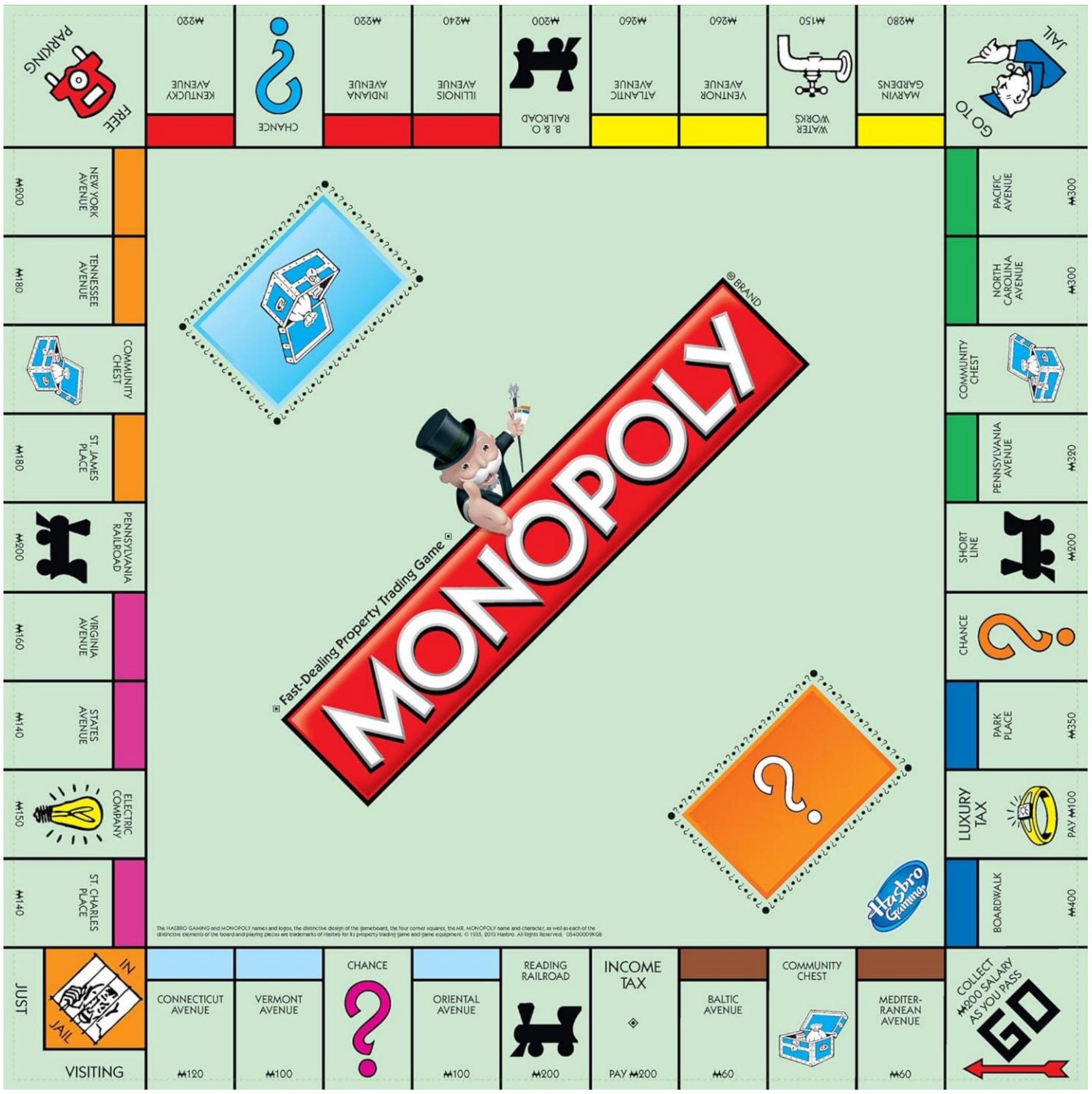
Hands-on Pretest Exam

Name: ______________________

Program and Year: ____________ **Date:** __________

SCORES		
Part 1		**70**
Part 2		**70**
Part 3		**60**
Part 4		**40**
Part 5		**90**
Total		**330**

THE MONOPOLY GAME (v2.0)



The story: Fast forward two years after he passed the practical exam for Programming 2 (which was about Monopoly), Mr. Moo Noo Po Lee just received his report card for first semester AY 2023-2024. Most of his grades were fine, except that his grade for a certain subject with course code "IDKWITHTHISANYMORE 11872638462" which translates to the subject "Easiest Practicum Course in the World" was worse than the grades of 47 other subjects that he took. After knowing that a lot of his classmates got a 1.0 on that same subject, he felt emotionally unstable. Here's a photo of Mr. Moo Noo Po Lee crying in pain after his shocking discovery of the grade drop.

When he lined up to enroll for his final semester in the DNDSA (Department of networking, data structures, and algorithms) office, his coordinator Ms. Khentintin Dela Pass (also the department's OJT coordinator) was not satisfied of the low OJT grade Mr. Moo Noo Po Lee has received as it was not well aligned with the performance during his initial years. So instead of actually catering his enrollment needs, the coordinator decided to just block him in ISMIS and give him additional tasks as a means to verify his skills before unblocking and enrolling him (that is, if there are still slots available by the time he submits the activities).

Mr. Moo Noo Po Lee was tasked to clean and organize the masterlist of the current freshmen students who are about to take the course "CIS 1201 – Programming 2" with nine assigned teachers handling the subject this semester – Christine, Pena, Bandalan, Gran, Sabandal, Moo, Noo, Poo, and Lee. He reached out to the president of the Computer and Information Sciences Council, Jose Lorraine Marie Tantay Montero, who in turn responded that the officers made a form via Google Forms that collates the students requesting for the said course to be offered.

The original masterlist contain the following details:

- ID number of the student
- Full name
- Program and year
- Latest programming 1 grade, as encoded in ISMIS

Sample data:

ID	**Name**	**Program and Year**	**Programming 1 Grade**
62349861238	Dayata, Wayne Matthew A.	BSCS-3	INC
124861281	Ivan Ric P. Woogue	BS CS 1	2.5
48293481237804	Gerard Ompad	BS Information Technology – 3	NC
21584688	Lim, Robert Michael	BSIT1	5.0

Mr. Moo Noo Po Lee seems annoyed looking at the list as the contents seem unorganized yet as the formatting is not consistent. He tries to contact the department's data science and data engineer instructors, Gerard and Ompad, respectively, to seek help in reorganizing the details, but sadly both of them are on leave until the end of the month.

Now he has no choice but to ask help from you (the person currently reading this) in automating the data cleaning task in which at the end, he needs to produce a cleaned masterlist in a CSV file containing only the students passing Prog 1.

Perform all the following tasks below. For each number, modify the code file (.c) in the given folder, then copy and save each output in the "Output" folder.

Program files that are not running due to any kind of error will not be checked, and a score of zero (0) will be immediately assigned for that section, so review your codes properly before submitting. Only correctly implemented functions will be awarded points which are indicated at the rightmost column of the table of functions.

Note: This shall serve as a practice for you to better prepare for the difficulty adjustments expected for Prog 2. You are free to search for additional resources to help you understand new concepts as highlighted. DO NOT ATTEMPT TO USE ANY AI TOOL TO DIRECTLY GENERATE OR ADJUST YOUR CODE, NOR ASK FOR ANSWERS DIRECTLY FROM ANY OTHER PEOPLE!

Task 1 (Number 1 – first folder):

- Objective: To arrange the masterlist by sorting it in increasing order of student IDs.
- Scenario: You are initially given a list of IDs to compare and arrange. Each ID is a string that consist of numeric characters and can contain several characters (up to 20 characters), and the IDs can be sorted the same way numbers are typically sorted (i.e. the equivalent value). For instance, "1567" is smaller than (and should go first before) "12345". In addition, all duplicate entries shall be removed from the list.
- Topics to research: Array manipulation, string functions (string.h library), sorting algorithms
- Implement the functions below:

Function header	Function definition	Points
`int length(char ID[])`	Returns the number of characters found in the given string representing a student ID. Example: "12345" returns 5.	4
`void swap(char ID1[], char ID2[])`	Swaps the two string contents provided i.e. ID1 will contain ID2 and ID2 will contain ID1 after the function completes its execution.	6
`int compare(char ID1[], char ID2[])`	Compares which of the given ID strings are greater based on the equivalent integer value. Returns 1 if the second value is greater, -1 if the first value is greater, and 0 if the two values are equal. This function must invoke length().	15
`void sortIDs(char*IDs[], int IDcount)`	Swaps the two string contents provided i.e. ID1 will contain ID2 and ID2 will contain ID1 after the function completes its execution. The number of IDs contained in the array can be found in the IDcount variable. This function must invoke compare() and swap().	15
`void displayIDs(char*IDs[], int IDcount)`	Display the list of IDs one by one. Each ID is printed in a single line. When called after the sortIDs(), the IDs displayed should be in sorted order. The number of IDs contained in the array can be found in the IDcount variable.	5
`int hasDuplicates( char*IDs[], int IDcount)`	Checks whether the list has duplicate entries. Return 1 if there are duplicates and 0 if there are none.	10
`void removeDuplicates( char*IDs[], int *IDcount)`	Removes all duplicate ID entries from the list and shifts the rest of the elements to their lower indices. The IDcount shall be modified accordingly based on the number of items removed from the array.	15
TOTAL POINTS (TASK 1)		**70**

Task 2 (Number 2 – second folder):

- Objective: To extract the different parts of a full name provided through varied formats.
- Scenario: You are initially given a list of full names which you have to identify the first name, middle initial (if it exists), and also last name. At the end, you should be able to transform a given list of full names into its organized format using a structure.
- Topics to research: Manipulating strings, structures, returning arrays from functions.
- Implement the functions below:

Function header	Function definition	Points
`int checkNameFormat( char name[])`	Checks if the given full name is written in last name first format or first name first format. Return 1 if the name is written starting with the last name, and 2 if the name is written starting with the first name.	5
`char getMiddleInitial( char name[])`	Retrieves the middle initial of a given full name. If there is no middle initial, return a null character.	5

`nameType getNamesLNFirst( char name[])`	Extracts the first name, middle initial, and last name of a given full name written in last name first format and places them accordingly in a nameType structure variable to be returned to the calling function. This function must invoke getMiddleInitial(). Note: Do consider that first names and last names can have more than one word! Example: Full name: Dela Paz, Heinz Lex Florenz P. Extracted first name: "Heinz Lex Florenz" Extracted last name: "Dela Paz" Extracted middle initial: 'P'	15	
`nameType getNamesFNFirst( char name[])`	Extracts the first name, middle initial, and last name of a given full name written in first name first format and places them accordingly in a nameType structure variable to be returned to the calling function. This function must invoke getMiddleInitial(). Note: If there is no middle initial, place the last word as the last name and the rest of the words as the first name. Example: Full name: Heinz Lex Florenz Dela Paz Extracted first name: "Paz" Extracted last name: "Heinz Lex Florenz Dela" Extracted middle initial: (none)	15	
`nameType getNameInfo( char name[])`	Determines the format of a given name using checkNameFormat() and calls the appropriate getName___() function to extract the names. Return the nameType structure variable obtained from either function to the calling function.	6	
`nameType* getNameInfoFromArray( char *name[], int count)`	Creates an returns an array (from the heap memory) of extracted name information from a given array of full names with mixed name formats. The number of full names is given in the count parameter. This function must invoke getNameInfo().	12	
`void displayName( nameType name, int format)`	Display on screen the full name from the given information in the name variable using the given format (1 – Last name first, 2 – First name first). Example: Last name first: Dayata, Wayne Matthew A. First name first: Wayne Matthew A. Dayata If there is no middle initial, do not print an extra period on the name.	6	
`void displayNamesFromArray (nameType names[], int format, int count)`	Display on screen a list of full names from the given array of name information using the given format. This function must invoke displayName(). Each full name is printed in a single line. The number of names in the array is given by count.	6	
TOTAL POINTS (TASK 2)		**70**	

Task 3 (Number 3 - third folder)

- Objective: To identify the program and year level from strings written in different program and year level formats.
- Scenario: When filling in the Google Form, most students tend to not follow the provided format in filling the program and year level field, resulting in these possible formats:
  - BS CS - 3
  - BSCS - 3
  - BS CS 3
  - BSCS 3
  - BS Computer Science 3
  - BS Comp Sci 3
  - Computer Science - 3
  - 3 Computer Science
- The resulting program shall be able to handle all those inconsistencies and transforms to the given formats. It can also be capable of modifying an existing program and year entry with the new inputs (in case a student likes to change program or updates his or her year level).
- Topics to research: Character functions, enumeration type (enums)
- Implement the functions below:

Function prototype	**Function definition**	Points	
`int getValueFromString( char string[])`	Extracts the digits from the given string to form an integer to be returned to the calling function. Example: Given: "Wa1ne Ma23hew A. Da4a5a" Value to return: 12345	15	
`program getProgram( char string[])`	Identifies the program in a given string (formats provided above) to be return to the calling function. Note: Use the constants, not the equivalent integer of the constants	25	
`progYearType getProgYear( char string[])`	Returns a progYearType structure containing the program and year. This function must invoke getValueFromString() to get the year level and getProgram() to get the program.	4	
`void editProgYear( progYearType *item, program prog, int year)`	Modifies a given progYearType item by assigning the prog and year variables to the structure's appropriate fields.	6	
`void displayProgYear( progYearType item)`	Displays the program and year level from the given program and year information. Example: Given values: BSCS, 3 Display: "BS Computer Science – 3"	6	
`void displayProgYearFromArray( progYearType items[], int count)`	Displays a list of program and year level information by invoking displayProgYear() per entry in the given array. Each program and year info is printed in a single line. The number of items in the array is given by count.	4	
`TOTAL POINTS (TASK 3)`		**60**	

Task 4 (Number 4 – fourth folder):

- Objective: To produce a consistent format of grades in the masterlist.
- Scenario: It has come to the department's attention that some teachers gave out wrong grades for Programming 1 that are not part of what has been agreed. Originally, the teachers are supposed to give numeric grades which can be either 1.0-3.0 (pass) or 5.0 (fail), where grades from 3.1-4.9 in the class record shall automatically result to 5.0. However, there are reports that "INC" and "NC" grades were given which are of different meanings to failed grades, which confused the students.
- Your task is to automate the conversion of all inappropriate grades (INC and NC) to 5.0 while also fixing occurrences of some failing grades beyond 3.0 that were not encoded by changing them as well to 5.0. Results are also converted to floating point data types instead of the original string data type. In addition, an extra field is to be created indicating whether the student has PASSED or FAILED the Programming 1 course.
- Topics to research: String manipulation, enums, display formatting
- Implement the following functions:

Function prototype	**Function definition**	Points
`int isNumeric( char grade[])`	Determines the kind of grade given and return 1 if the given grade is in numeric format and 0 if otherwise (such as "NG", "NC, "INC")	6
`float convertToFloat( char value[])`	Converts a decimal number from string format to float and returns the result to the calling function. Example: "2.5" returns 2.5, "4.56" returns 4.56. Assume that all the inputs are in valid format.	12
`float convertGrade( char grade[])`	Converts a given grade in string to its equivalent float value following the specifications given in the problem (convert 3.1-4.9 and letter grades to 5.0 and the rest as is). Assume that all the inputs are in their valid ranges (no grade beyond the ranges 1.0 and 5.0 can exist). This function must invoke isNumeric() and convertToFloat().	8
`status getStatus( float grade)`	Determines and returns the status of a given numeric grade as either PASSED or FAILED.	3
`gradeType getGradeType( char grade[])`	Creates and returns a gradeType variable populated with appropriate values in the fields by invoking convertGrade() and getStatus().	5
`void displayGradeType( gradeType grade)`	Displays the grade and status from the given gradeType information. Example values: 3.0, PASSED Display: "3.0 - PASSED"	3
`void displayGradeTypeFromArray( gradeType items[], int count)`	Displays a list of grade and status information by invoking displayGradetype() per entry in the given array. Each grade and status info is printed in a single line. The number of items in the array is given by count.	3
`TOTAL POINTS (TASK 4)`		**40**

Task 5 (Number 5 – fifth folder):

- Objective: To output a CSV file containing the cleaned data by applying the programs for cleaning the different areas of the masterlist from tasks 2 to 4.
- Scenario: Now that you have finished creating the functions that help transform the different parts of the data, it is now time to put them together to see the final result. If your implementations from the first four tasks are all flawless, then there should be no further issues going into this task as this will be focused on file handling, from importing the data, applying the data cleaning functions, to generating and exporting a new CSV file containing the refurbished masterlist!
- As an additional requirement, you are to also generate a true, official masterlist containing only the students that passed Programming 1 (since this is the prerequisite course for Programming 2) – remember, we need to do proper data validations too!
- Before starting, be sure to paste the codes from all the functions you created in tasks 2 to 4 into the program file provided for this task.
- Topics to research: ADT List (array implementation), passing structures by copy and address/reference, File handling (reading from and writing to a file)
- Implement the functions below:

Function prototype	**Function definition**	Points	
`void initOldList( oldListType *oldList)`	Initializes the old list by setting the count field to 0.	1	
`void initNewList( newListType *newList)`	Initializes the new list by setting the count field to 0.	1	
`void importOldList( char fileName[], oldListType *oldList)`	Opens a CSV file with the given file name (file format included) then populates the information into an oldListType list in memory. After populating the list, the file shall be closed and the resulting list shall be returned to the calling function.	12	
`void cleanData( oldListType oldList, newListType *newList)`	Invokes all the data cleaning functions (name, program and year, and grade) by passing in information from the old list and storing the results into the given new list.	20	
`void sortListByLastName( newListType *newList)`	Sorts the given new list by increasing alphabetical order of last name. If the last names are same, then compare using the first names.	16	
`void displayGradeStatusCounts( newListType newList)`	Display the count of students that passed and failed the Programming 1 subject, as well as the percentage of students passed, using the format below. Display format: Passed students: 50 (50%) Failed students: 50 (50%)	10	
`newListType getPassedStudentsFromList( newListType List)`	Identifies the students that passed programming 1 based on the grade status and copies them into a new list to be returned to the calling function.	15	
`void addStudentsToFile( char fileName[], newListType List)`	Creates a new CSV file with the indicated file name (file format included) and populates all the information from the newList (cleaned data) into the file. Refer to the program file for the formatting.	15	
`TOTAL POINTS (TASK 5)`		**90**	