BITS Pilani Pilani | Dubai | Goa | Hyderabad	**Work Integrated Learning Programmes Division** **M.Tech (Software Engineering)**
(S2-21_SESAPZG519) **(Data Structures and Algorithms Design)** **Academic Year 2021-2022**	
Assignment 1 – PS5 – [To Do List] - [Weightage 12%]	

1. Problem Statement

Your goal is to create a to-do-list and associated operations using linked-lists in python.

The To-Do list should have the capability of:

1. Adding/Creating a task
2. Assigning a task-Id
3. Removing a task
4. Searching a task
5. Marking a task completed
6. Unmarking a task as completed
7. List status showing all completed and incomplete tasks and associated details.

To solve this problem, create a linked-list of tasks and mark them as complete and incomplete.

You'll will be given few commands in the input file.

Operations:

1. **def initiateToDoList(read_input_file):** This function reads the input file and creates a to-do list and all associated data structures and calls the necessary functions as mentioned in the input file.
   Input: Input-File name with path.
   Output: None

2. **def addTask(task_string=""):** This function is called by initiateToDoList and creates a task along with a unique task-number and writes this task-number in the output file.
   Input : Task string.
   Output: "ADDED:"<Unique Task-Number assigned to this task.>-<task-string>

3. **def removeTask(task_string="", task_number=""):** Description: This function is called by initiateToDoList and removes a task along with a unique task-number and writes this task-number in the output file.
   Input : Task string. OR task_number.

Output: “REMOVED:”<Unique Task-Number assigned to this task.>-<task-string>

4. **def searchTask(search_string=””):** This function is called by initiateToDoList and find a task along with a unique task-number and writes this task-number in the output file.
   Input : Task string. OR task_number.
   Output: SEARCHED:<search-string>

   <Task-Number>-<task-string>
   ---

5. **def completeTask(task_string=””, task_number=””):** This function is called by initiateToDoList and marks a task along with a unique task-number as complete and writes this task-number in the output file.
   Input : Task string. OR task_number.
   Output: “COMPLETED:”<Unique Task-Number assigned to this task.>-<task-string>

6. **def incompleteTask(task_string=””, task_number=””):** This function is called by initiateToDoList and unmarks a task along with a unique task-number as complete and writes this task-number in the output file.
   Input : Task string. OR task_number.
   Output: “UNCOMPLETED:”<Unique Task-Number assigned to this task.>-<task-string>

7. **def statusTask():** This function is called by initiateToDoList and unmarks a task along with a unique task-number as complete and writes this task-number in the output file.
   Input :
   Output: TASK STATUS:

   <Task-Number assigned to this task.>-<task-string>

Requirements:

1. Implement the above problem statement using **Python 3.7**
2. Read the input from a file(**inputPS5.txt**), which contains the list of tasks and associated actions to be taken identified by relevant tags at the start of each line separated with a colon.
2. You will output your answers to a file (**outputPS5.txt**) for each line.
3. Perform an analysis for the features above and give the running time in terms of input size: n.

Sample Input:

Input will be taken from the file(**inputPS5.txt**).

```
Add a Task: Complete Assignment.
Add a Task: Return Library Books.
Add a Task: Book a ticket.
Add a Task: Wish Birthday
Remove Task: Wish Birthday
Mark Complete: Complete Assignment.
Task Status:
Mark InComplete: Complete Assignment.
Remove Task: Complete Assignment.
Add a Task: Call Home.
Add a Task: Read Algorithm book
Remove Task: Call Home.
Search Task: book.
Add a Task: wear mask,
```

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.

Sample Output:

Display the output in **outputPS5.txt**.

```
ADDED:TA3291-Complete Assignment.
ADDED:TA8920-Return Library Book.
ADDED:TA4836-Book a ticket.
COMPLETED:TA3291-Complete Assignment.
TASK-STATUS:
Task-Number   Task-String              Task-Status
-----------------------------------------------------
TA3291        Complete Assignment.     C
TA8920        Return Library Book.     I
TA4836        Book a ticket.           I
-----------------------------------------------------
UNCOMPLETED:TA3291-Complete Assignment.
REMOVED:TA3291-Complete Assignment.
ADDED:TA3111-Call Home.
ADDED:TA3916-Read Algorithm book
REMOVED:TA3111-Call Home.
SEARCHED:book
-------------------------------
TA8920-Return Library Book.
TA4836-Book a ticket.
---------------------------
ADDED:TA9374-wear mask,
```

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.

2. Deliverables

1. Word document **designPS5_<group id>.docx** detailing your design and time complexity of the algorithm.
2. **[Group id]_Contribution.xlsx** mentioning the contribution of each student in terms of percentage of work done. Download the Contribution.xlsx template from the link shared in the Assignment Announcement.
3. **inputPS5.txt** file used for testing
4. **outputPS5.txt** file generated while testing
5. **.py file** containing the python code. Create a single *.py file for code. Do not fragment your code into multiple files

Zip all of the above files including the design document and contribution file in a folder with the name:

[Group id] _A1_PS5_ ToDoList.zip and submit the zipped file.

Group Id should be given as **Gxxx** where xxx is your group number. For example, if your group is 26, then you will enter G026 as your group id.

3. Instructions

1. It is compulsory to make use of the data structure(s) / algorithms mentioned in the problem statement.
2. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full. Also ensure basic error handling is implemented.
3. For the purposes of testing, you may implement some functions to print the data structures or other test data. But all such functions must be commented before submission.
4. Make sure that your read, understand, and follow all the instructions
5. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.
6. The input, prompt and output samples shown here are only a representation of the syntax to be used. Actual files used to evaluate the submissions will be different. Hence, do not hard code any values into the code.
7. Run time analysis is to be provided in asymptotic notations and not timestamp based runtimes in sec or milliseconds.

Instructions for use of Python:

1. Implement the above problem statement using Python 3.7.
2. Use only native data types like lists and tuples in Python, do not use dictionaries provided in Python. Use of external libraries like graph, numpy, pandas library etc. is not allowed. The purpose of the assignment is for you to learn how these data structures are constructed and how they work internally.
3. Create a single *.py file for code. Do not fragment your code into multiple files.
4. Do not submit a Jupyter Notebook (no *.ipynb). These submissions will not be evaluated.
5. Read the input file and create the output file in the root folder itself along with your .py file. Do not create separate folders for input and output files.

4. Deadline

1. The strict deadline for submission of the assignment is **10th April 2022.**
2. The deadline has been set considering extra days from the regular duration in order to accommodate any challenges you might face. No further extensions will be entertained.
3. Late submissions will not be evaluated.

5. How to submit

1. This is a group assignment.
2. Each group has to make one submission (only one, no resubmission) of solutions.
3. Each group should zip all the deliverables in one zip file and name the zipped file as mentioned above.
4. Assignments should be submitted via E-Learn > Assignment section. Assignment submitted via other means like email etc. will not be graded.

6. Evaluation

1. The assignment carries 12 Marks.
2. Grading will depend on
    a. Fully executable code with all functionality working as expected
    b. Well-structured and commented code
    c. Accuracy of the run time analysis and design document.
3. Every bug in the functionality will have negative marking.
4. Marks will be deducted if your program fails to read the input file used for evaluation due to change / deviation from the required syntax.

5. Use of only native data types and avoiding libraries like numpy, graph and pandas will get additional marks.
6. Plagiarism will not be tolerated. If two different groups submit the same code, both teams will get zero marks.
7. Source code files which contain compilation errors will get at most 25% of the value of that question.

7. Readings

Text book: Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition). **Chapters:** 2.2