Mini-Project

ET0702 2022_23 Sem 2

Grouping
Group size: Minimum 2 and maximum 3 students.

Expectations
Each group is expected to do requirement analysis, system design, implementation, and finally a demonstration and a report of the software.

This is not a one-weekend task. You are expected to spend time discussing, trying codes out, modifying, and improving your results and user experience. As you work along, you are also encouraged to discuss with your lecturers about issues, your approaches, clarifying the needs and expectations while showing your progress in the project through demonstration.

You are required to show/discuss with your lecturer on a regular basis the progress of your work. Marks will be awarded for novelty of implementation and on improvement on requirement not expressly stated. Early submission is strongly encouraged and will be rewarded with ***marks***.

Item	**Marking Criteria**	**Max Marks**
Individual marks		
1	Quality of individual contributions to project in terms of: * research and testing etc. [5] * designing, evaluating and selecting algorithms. [15] * designing of data structures [10] * coding [20]	50
2	Final presentation and demonstration	10
3	Interview Q&A	10
Group marks		
4	Regular updates and demo of the contributions in discussions and coding in progress	10
5	Short reflective report on the strategies and development experience, Source Code Documentation & User Guide (hard copy)	10
6	Early submission & presentation (For presentation in week 17 [6-10 Feb 2023], in respective timetabled practical session)	10

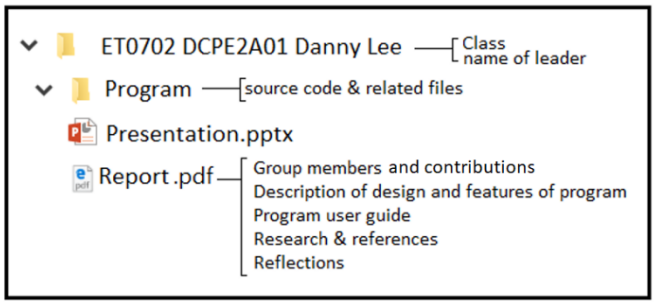
Topic

You can choose or be assigned to any of the topics listed. You may also select to do a topic that interest you and of your own conception, with your lecturer's approval.

Submission

Final submission - Week 18, on respective timetabled practical/tutorial lesson.
(Early submission marks for Week 17(or earlier)
This assignment constitutes 20% of the total assessment for this module.
Your deliverables should include:

- A short presentation and Q&A session
- A softcopy of the following items in the folder structure as shown below:



Sloppy report, without proper member identification or any non-conformance to the folder structure specified will suffer downgrade.

The purpose is to identify the name and admission number of each member and record the contributions of each member in the group.

Report – *Description of design and features of the program*

The purpose is to enable the succeeding programmers to be able to have a clear understanding on the inner working of the software at a glance. The document should contain the requirements of the project, analysis as reflected by the problem given, and continue with explanations for the chosen design of the software, all functions that were coded, brief description on the purpose of the function, variables used by the function and also any global variables declared in the software.

You are free to use any available libraries, but you must explain what it is and what you are using the library for. Operating environment, such as the hardware platform, the type of development environment and the running environment should be stated.

Report – P*rogram User Guide*

The purpose is to aid the users on how to use the software. This guide can be purely instructional, with screen shots directing the users and should be simple and intuitive. For the developer, the user guide is a reminder to the need to develop simple and intuitive operations.

Notes on Topics and Mini-project

The project descriptions of the different topics below are not exhaustive and are never intended to be so.

Real problems and issues always begin with vague, ambiguous and open-ended needs. Only as the problem is explored with questions, analyses and counter suggestions, only then issues can be clarified, needs understood and implementations proposed.

Some stated requirements may even be renegotiated and changed as a result of better understanding of the problems and their changed contexts. This incompletion of the project description will give room for students and lecturers to interpret and evolve useful and practical ideas from them as they consider choices of approaches, strategies as well as usage requirement from the view point of a user.

All eventual forms of software applications arise from the cut-and-thrust of discussions, negotiations and refinements of ideas. This exercise will not try to restrict this natural process of evolvement.

It is important for you to test your codes and discuss your approaches with your lecturer early to avoid misinterpretations or erroneous/unrealistic/overly simplistic assumptions.

You are encouraged to discuss with your lecturer early and progressively.

You are advised to conduct a research on the chosen topic to learn about the central and peripheral issues involved.

Topic 1 – Storage Rack Management System

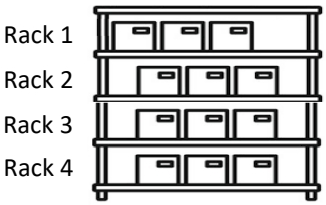
A rented storage has racks for storing packages of goods. Each rack has a different of weight limits and cost (to be read in from a text file). Assuming that there are only 2 types of packages (of different weights).

Develop a program to read in, from file, the number of racks, weight limits and costs of the racks, the weights of the 2 different types (assuming always 2 types only) of packages (not necessary must be 2kg and 3kg as long as they are different) and the number of packs for each type of packages. Feel free to design any file formats.

The program then prints out the most cost effective arrangement of the packages on the racks and other useful information. Assuming that a rack will be charged as long as the rack is being used (even if it is not fully packed to the limit)

An example:

- 4 racks
- Packages: 2kg and 3kg
- 4 packs of 2kg
- 3 packs of 3kg



Output (just a guide):

```
Number of 2kg: 4 packs
Number of 3kg: 3 packs

Rack 1 ($0)
================================================
Rack 2 ($0)
================================================
Rack 3 ($9.60)    2kg  2kg  3kg              (wastage: 1kg space)
================================================
Rack 4 ($10.00)  2kg  2kg  3kg  3kg
================================================

Total costs: $19.60
Storage wastage: 1kg
Outstanding packages (racks are full): 0
```

Begin the project with a smaller number of racks and packages. Figure out the data structures and algorithms for the storage plan without any optimisation. Next, increase the numbers and enhance the algorithm.

[For this assignment, limit to maximum 4 racks]

Program must include: functions, array, file access and list/queue/vector and flexible enough to take different combination of input data (not limited to the example shown above).

Topic 2 – FC3 Western Food Cooking Planning System

This program aims to improve the productivity of the FC3 Western Food kitchen and capitalise on economic of scale.

Cooking a dish one at a time is costlier than cooking in a batch.
However, batch cooking has limit due the capacity of the kitchen.
Assuming that the kitchen can only cook 1 type of dish at any one time. However, multiple servings (depends on the limit) of the dish can be cooked concurrently.

Develop a program to read in, from file, the batch limits of each dish and the order information. Feel free to design the file formats. The program then prints out the cooking plan, serving the orders according to the input sequence.

An example:

- 4 dishes (with maximum limits)
- 3 orders

Dish	maximum/batch
A	3
B	2
C	3
D	4

	quantity			
Order	A	B	C	D
1	0	0	0	2
2	1	1	0	1
3	1	0	1	2

Assuming that in optimised cooking, when planning the cooking of a dish (e.g. say D in Cook# 1 in above example), the program will scan all the others orders having dish D (Order# 1,2, and 3) until the limit is hit. If the capacity cannot fulfil the quantity of a dish required by an order (Order# 3 needs D x2), the cooking of the dish (D x2) for this particular order (Order# 3) will have to wait.

Output (just a guide):

```
Cook 1
======
Order 1 – dish D x2
Order 2 – dish D x1
Completed: Order1

Cook 2
======
Order 2 – dish A x1
Order 3 – dish A x1

Cook 3
======
Order 2 – dish B x1
Completed: Order 2

Cook 4
======
Order 3 – dish C x1

Cook 5
======
Order 3 – dish D x2
Completed: Order 3

Total cooking: 5
Total dishes: 9
Total A B C D: 2 1 1 5
```

Begin the project with a smaller number of dishes and order, without optimisation and no limits on the dishes. Figure out the data structures and algorithms for the cooking plan. Next, increase the numbers and enhance the algorithm.

[For this assignment, limit to maximum 4 dishes and maximum 5 orders]

Program must include: functions, array, file access and list/queue/vector and flexible enough to take different combination of input data (not limited to the example shown above).

Topic 3 – Golf Club Delivery System

This program aims to generate a delivery plan for an autonomous delivery vehicle operating in a golf country club between predetermined locations. The distance between various locations in the club is predetermined.

Initially, the vehicle is at the home location. This is also the location that the vehicle will return to after completing all the job orders. A job order involves delivering stuff from one location to another location.

Develop a program to read in, from file, the number of locations, distances and job orders. Feel free to design the file formats. The program then prints out the delivery plan, preferably with optimised distance travelled.

Assuming that the vehicle can pick up unlimited job orders from a location.

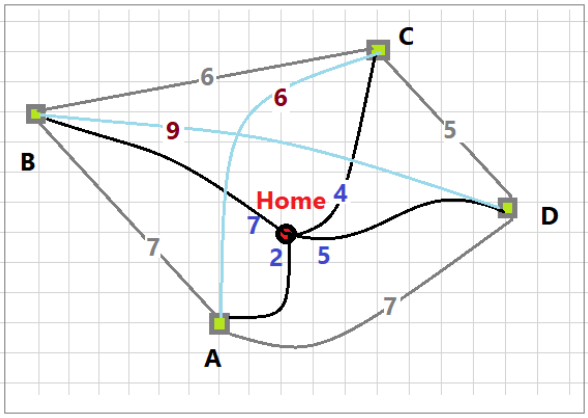
Begin the project with a smaller number of locations, say 2 locations. Figure out the data structures and algorithms for the delivery plan without optimising the travelling time. Next, increase the numbers and enhance the algorithm.

[For this assignment, limit to maximum 4 locations and maximum 7 jobs]

An example:

- 4 locations and the distances (to between them
- 6 job orders

Job	From	To
1	B	D
2	A	C
3	C	B
4	A	B
5	D	A
6	C	B



Output (just a guide):

Route 1: Home to A (2)
Distance travelled: 2

Route 2: A to C (6)
Distance travelled: 8
Job 2 completed

Route 3: C to B (6)
Distance travelled: 14
Job 3 completed
Job 6 completed

Route 4: B to D (9)
Distance travelled: 23
Job 1 completed

Route 5: D to A (7)
Distance travelled: 30
Job 5 completed

Route 6: A to B (7)
Distance travelled: 37
Job 4 completed

All jobs completed.
Route 7: B to home (7)
Distance travelled: 44

Program must include: functions, array, file access and list/queue/vector and flexible enough to take different combination of input data (not limited to the example shown above).