Assignment 2

Aim

The objectives of this assignment includes:

- Learning about encapsulation, inheritance, polymorphism and function overloading
- Apply the concepts learnt by developing a survey and path planning program

Background

In a theoretical flat-land universe, everything is in 2 dimensions. People, animals, plants to planets, moons, galaxies and even space itself, is in 2D. In our flat-land space (i.e. 'flat-space'), there is a powerful organization called 2D-StarFleet (2DSF), whose goals include seeking out new life and civilization via exploration.

While on a routine mission of exploration, the flagship of 2DSF, the Enterprise-2D is trapped in an expanse of space encircled by a massive ring of violent, electrical plasma storm. Data coming in from the sensor array reveals that the only opening in this storm is located at the far end of the enclosed area, from Enterprise-2D's current location.

In addition, the sensor data also revealed that this area is populated by strange, 2D geometrical shapes, with sizes ranging from a small moon, asteroid, to large planets, or even a star! This implies that to travel to the 'exit' at the far end of the storm, you need to understand more about the properties of these shapes and attempt to chart a course to navigate to the exit!

As a Science Officer aboard Enterprise-2D, you need to develop a program that has the following capabilities:

a) read in sensor data on the strange 2D shapes (via manual input)
b) compute the area ('mass') of these shapes
c) print shapes report (e.g. list of points: on its perimeter, or totally within shape's area)
d) sort shapes data (sorted by special type and area)

The next section provides information about the requirements for developing this program.

Task Requirements

A) In terms of relative positioning, you may assume a coordinate system with Enterprise-2D at the origin, trying to navigate in a general 'upper-right' direction, to get to the exit in the storm. Please refer to **Appendix A**, which elaborates on this coordinate system and the unit representation of 2D shapes.

IMPORTANT : For this assignment, you should not assume that the 2D shapes in **Appendix A** are positioned exactly as shown in **Appendix A**, nor that there are not more shapes. There will, however, only be shapes of the types listed in **Appendix B**

B) The sensor data coming in from Enterprise-2D's sensor array provides crucial information about the 2D shapes such as name, special type and location of all vertices (that outlines the perimeter of the shape). Please refer to **Appendix B**, which provides a more detailed description of the sensor data.

C) To assist you in the initial class design of your program, please refer to **Appendix C**, which illustrates one possible way of designing your program. It also describes a list of requirements which you need to implement, especially those marked under "compulsory". The classes highlighted in **Appendix C** are purely meant to store data about the 2D shapes entered into your program by user.

D) You are required to implement a main driver file called '`Assn2.cpp`', whose methods are called to start the program. When started, it should print a menu providing the following functionalities :

- read in sensor data on the strange 2D shapes (via manual input)
- compute the area ('mass') of these shapes
- print shapes report (e.g. list of points on its perimeter, or totally within shapes area)
- sort shapes data (sorted by special type and area)

Appendix D provides more information about implementing this class.

E) Once the program is completed and tested to be working successfully, you are highly encouraged to add on "new features" to the program that you feel are relevant to the problem. Additional marks may be awarded subject to the relevancy and correctness of the new functionalities. (Note : the additional features will only be considered IF the program has correctly fulfilled all the basic requirements elaborated in the earlier sections!)

F) You are to use only C++ language to develop your program. There is no restriction on the IDE as long as your source files can be compiled by g++ compiler (that comes packaged in Ubuntu linux) and executed in the Ubuntu terminal shell environment.

Deliverables

1) The deliverables include the following:

   a) The actual working C++ program (soft copy), ***with comments on each file, function or block of code*** to help the tutor understand its purpose.

   b) A softcopy word document that elaborates on:
      - (Interpreted) requirements of the program
      - Diagram / Illustrations of program design
      - Summary of implementation of each module in your program
      - Reflections on program development (e.g. assumptions made, difficulties faced, what could have been done better, possible enhancements in future, ***what have you learnt***, etc)

   c) A program demo/software testing during lab session. You must be prepared to perform certain tasks / answer any questions posed by the tutor.

2) IMPT: Please **follow closely**, to the submission instructions in **Appendix E**, which contains details about what to submit, file naming conventions, when to submit, where to submit, etc.

3) The software demo / testing will be held during lab session where you are supposed to submit your assignment. Some time will be allocated for you to present / demonstrate your program's capabilities during the session.

Grading

Student's deliverable will be graded according to the following criteria:

(i) Program fulfills all the basic requirements stipulated by the assignment

(ii) Successful demonstration of a working program, clarity of explanation / presentation and satisfactory answers provided during Q & A session.

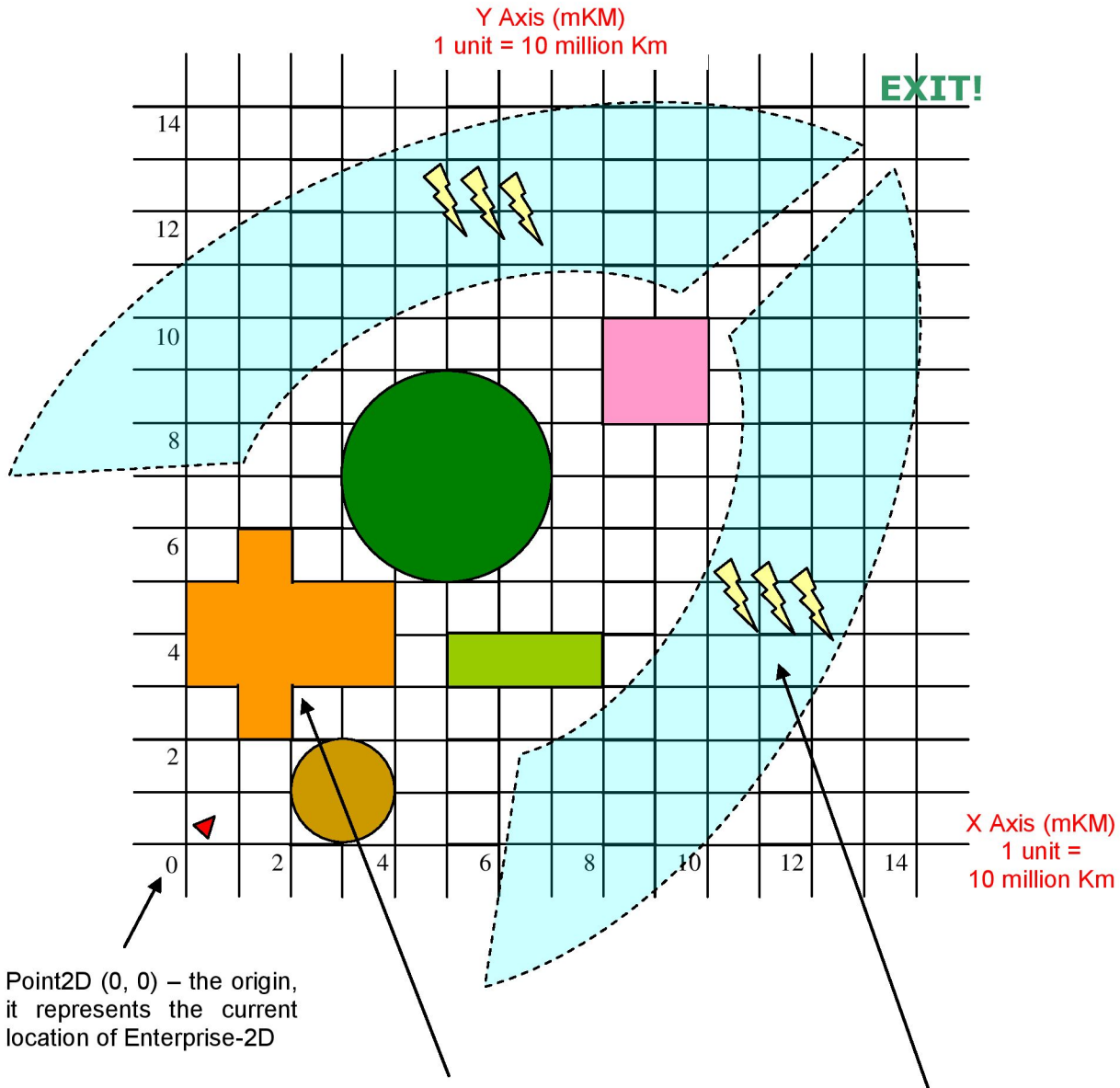
(iii) Additional effort (e.g. enhancing the program with *relevant* features over and above task requirements, impressive, 'killer' presentation)

(iv) After the submission of deliverables, students will be required undergo a software testing process (to determine the correctness and fulfillment of software requirements.) Further instructions will be given by the Tutor during the subsequent respective labs. Please pay attention as failure to adhere to instructions will result in deduction of marks.

Tutor's note:
In the real working world, satisfactory completion of your tasks is no longer enough. The capability, efficiency and robustness of your system to operate under different testing conditions, and the ability to add value, communicate and/or demonstrate your ideas with clarity is just as important as correct functioning of your program. The grading criteria is set to imitate such requirements on a 'smaller scale'.

APPENDIX A

(Coordinate System w.r.t. Enterprise-2D, and the plasma storm)



APPENDIX B

(Description of Sensor Data)

Name
The name of the 2D shape reveals the general type of shape encountered. Currently, the values consist of : "Square", "Rectangle", "Cross" and "Circle".

Special Type
Enterprise-2D's sensor has detected that some shapes encloses a 'warp-space' with the amazing ability to teleport any objects that touches one of its vertex, to any other vertices of the same shape, ***instantaneously*** !

This makes it highly desirable, for Enterprise-2D to travel towards this kind of shape, in the hopes of travelling faster, and saving precious fuel at the same time!

There are only 2 values for 'special type' : "WS" (Warp-Space) or "NS" (Normal-Space).

Vertices
The vertices is actually a **set** of (x, y) points, that describes the outline of the 2D shape. The number of points in the **set**, depends on the name of the shape. The table below summarizes the kind of sensor data your program expects to receive.

Name	Special Type	No. of Vertices (i.e. x, y, points)	Actual Vertex Data …
"Cross"	"WS" or "NS"	12	e.g. (1, 3), (1, 4), … etc.
"Square"	"WS" or "NS"	4	
"Rectangle"	"WS" or "NS"	4	

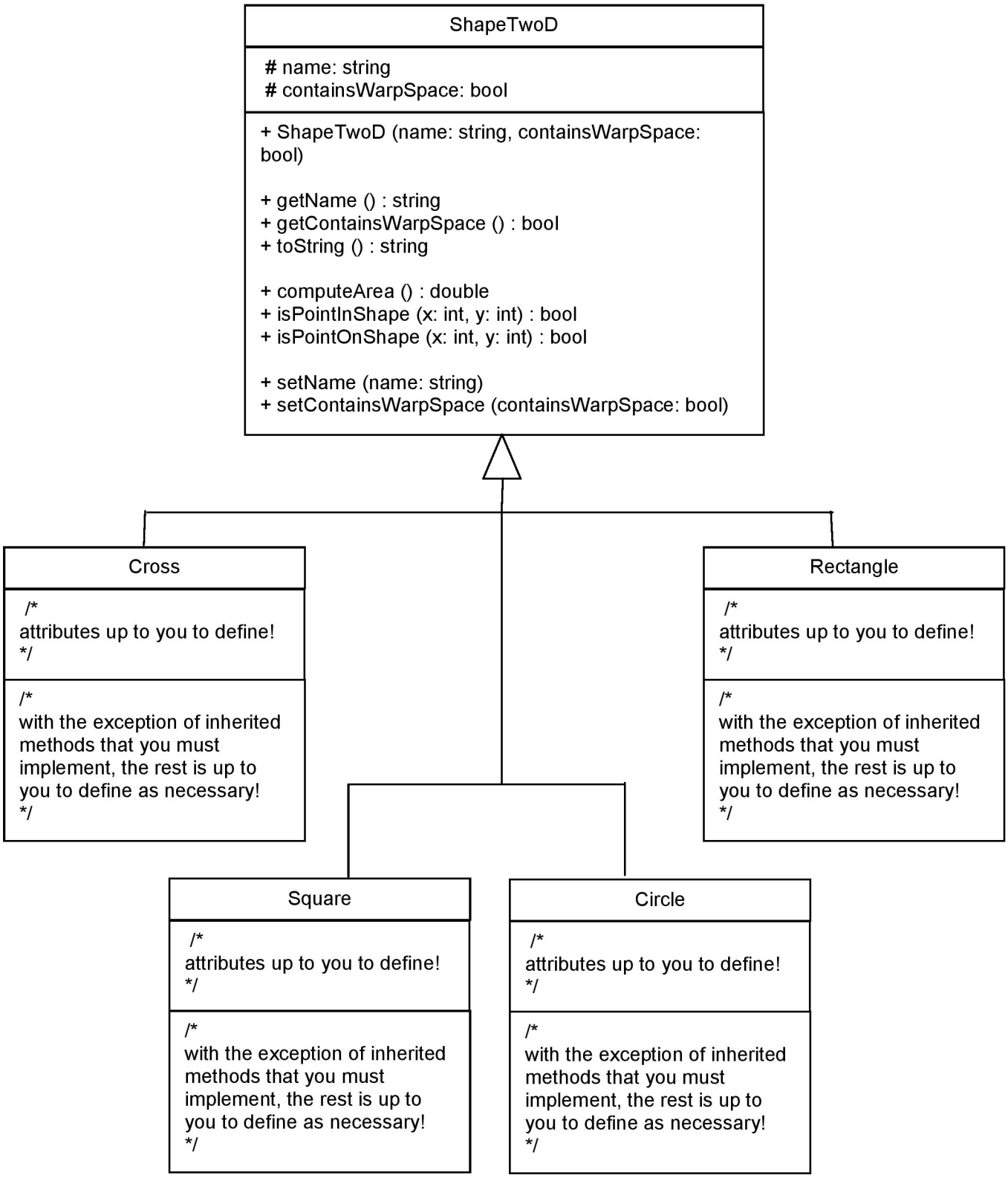
Note :
As mentioned in the Background section, the 1st capability of your program should allow manual input of the above data.

It is **not necessary** to prompt user for "No. of Vertices" because the name of the shape will already inform your program about how much vertex data to expect.

For example, when your program prompt for name of the shape, if user enters "Cross", it is safe for your program to assume that user is going to key a set of 12 (x, y) points later!

APPENDIX C

(Implementation Requirements)



Compulsory requirements

#1 The parent class is '`ShapeTwoD`'. Any attributes, constructors and methods specified in the diagram must be implemented, with the exact same name, parameter, type and access!

#2 The sub-classes of `ShapeTwoD` must be named '`Cross`', '`Square`', '`Circle`' and '`Rectangle`'.

#3 The method '`toString()`' in class `ShapeTwoD` is a virtual function that returns a string containing all the values of the attributes in the shape, excluding the array of vertex data. (However, sub-classes of `ShapeTwoD` must output the array of vertex data, inclusive of any other attribute's values it inherited)

#4 The method '`computeArea()`' in class `ShapeTwoD` is a virtual function. It must be override by the sub-classes and implemented individually.

For example, because each sub-class has different number of vertices and values, sub-class `Cross`'s `computeArea()` implementation would use a different algorithm / formula from sub-class `Square`'s `computeArea()` implementation!

Note : The sensor data will only provide the locations (vertices) of each 2D shape encountered. You will be required to do the necessary research to derive the formula to compute the area for each shape, based on the set of vertex data (i,e. a set of [x, y] points) provided!

#5 The method '`isPointInShape()`' in class `ShapeTwoD` is a virtual function. It takes in a [x, y] location and returns a boolean value indicating whether the location is **totally within** the shape's area. It must be over-ridden by the sub-classes and implemented individually. (PIs refer to sample output in **Appendix D**).

#6 The method '`isPointOnShape()`' in class `ShapeTwoD` is also a virtual function. It takes in a [x, y] location and returns a boolean value indicating whether the location is **found on** any lines joining the shapes' vertices! It must be over-ridden by the sub-classes and implemented individually. (PIs refer to sample output in **Appendix D**).

Other requirements

i) For parent class '`ShapeTwoD`'. You are free to add on any additional methods or attributes deemed necessary for your program to provide its services.

ii) Since user will key in the name of the shape, it is possible for you to declare arrays of fixed sizes in the sub-classes to store the coordinate vertices of the shape!

iii) You are free to implement any other additional classes you feel necessary so as to provide the required capabilities for this program.

APPENDIX D

(Implementation info for : `Assn2.cpp` driver file)

This file contains the **main ()** method which declares and instantiates all other classes (i.e. `Shape2D`, `Square`, … etc) and sets up all the necessary interactions to perform its task.

Student ID : 1234567
Student Name : Tan Ah Meng Elvis

Welcome to Assn2 program!

1) Input sensor data
2) Compute area (for all records)
3) Print shapes report
4) Sort shape data

Please enter your choice : **1**

[Input sensor data]
Please enter name of shape : **Cross**
Please enter special type : **WS**

Please enter x-ordinate of pt.1 : **1**
Please enter y-ordinate of pt.1 : **1**

. . .

Please enter x-ordinate of pt.12 : **1**
Please enter y-ordinate of pt.12 : **2**

Record successfully stored. Going back to main menu …

Student ID : 1234567
Student Name : Tan Ah Meng Elvis

The figure on the left describes a sample interaction between the main menu and '*Input sensor data*' sub-menu (1st example)

Note :
All shapes data should be stored in a `Shape2D` array in the `Assn2.cpp` file. You may assume that no more than 100 shapes will be entered into your program at any one time!

Impt !! Please include your:
1. *Student ID*
2. *Student Name*

every time you display your Main Menu. Marks will be deducted if the required info is not shown.

Student ID : 1234567
Student Name : Tan Ah Meng Elvis

Welcome to Assn2 program!

1) Input sensor data
2) Compute area (for all records)
3) Print shapes report
4) Sort shapes data

Please enter your choice : **2**

Computation completed! (5 records were updated)

The figure on the right describes a sample interaction between the main menu and '*Compute area (for all records)*' function.

Note : The example assumes the case where there are only 5 shapes input so far, hence, the message that '*5 records were updated*'!

In this compute area function, you must exhibit polymorphic behavior and dynamic binding by invoking the correct function for each shape stored in the `Shape2D` array !

Student ID : 1234567
Student Name : Tan Ah Meng Elvis

Welcome to Assn2 program!

1) Input sensor data
2) Compute area (for all records)
3) Print shapes report
4) Sort shape data

Please enter your choice : **1**

[Input sensor data]
Please enter name of shape : **Circle**
Please enter special type : **NS**

Please enter x-ordinate of center : **5**
Please enter y-ordinate of center : **7**
Please enter radius (units) : **2**

Record successfully stored. Going back to main menu …

Student ID : 1234567
Student Name : Tan Ah Meng Elvis

The figure on the left describes a sample interaction between the main menu and '*Input sensor data*' sub-menu (2nd example)

Note : For circle, there is no need to enter vertices, as there are no "corners" as opposed to other multi-sided polygon shape. Instead, just prompt user to enter the [x, y] of its center and its radius, as shown on the left …

The figure on the right describes a sample interaction between the main menu and '*Print shapes report*' function.

Notes : The example assumes the case where there has only been 5 sensor data records input so far.

'Points on perimeter' refers to only those points lying on the line drawn between 2 vertices, for each pair of vertices defining the shape's outline.

E.g. (1, 2) lies on a line between vertices (1, 1) and (1, 3) !

You do not need to include those points which describe the vertices of the shape!

'Points within shape' refers to those points which are **totally within** the shape.

You do not need to include :

- those points which describe the vertices of the shape!
- those points on the perimeter of the shape!

```
Student ID      : 1234567
Student Name  : Tan Ah Meng Elvis
-----------------------------------------------------
Welcome to Assn2 program!

1)      Input sensor data
2)      Compute area (for all records)
3)      Print shapes report
4)      Sort shapes data

Please enter your choice : 3

Total no. of records available = 5

Shape [0]
Name    : Square
Special Type : WS
Area : 4 units square
Vertices :
Point [0] : (1, 1)
Point [1] : (1, 3)
Point [2] : (3, 3)
Point [3] : (3, 1)

Points on perimeter : (1,2), (2, 1), (2, 3), (3, 2)

Points within shape : (2, 2)

. . .

Shape [4]
Name    : Rectangle
Special Type : NS
Area : 8 units square
Vertices :
Point [0] : (2, 17)
Point [1] : (2, 15)
Point [2] : (6, 15)
Point [3] : (6, 17)

Points on perimeter : (2, 16), (3, 15), (4, 15), (5, 15), (6,
16), (5, 17), (4, 17), (3, 17)

Points within shape : (3, 16), (4, 16), (5, 16)
```

Note for Circle :

Points on perimeter : only output the 4 points that are "North" / "South" / "East" / "West" w.r.t. the circle's center (x, y)

Points within shape : output all (x, y) points that are totally within the circumference. It is not compulsory to output the circle's center.

The figure on the right describes a sample interaction between the main menu and '*Sort shapes data*' sub-menu.

Note : For 'sort by special type and area' option in the sub-menu, shapes with special types 'WS' should be displayed first, followed by all shapes with special types 'NS'.

Within each group of shapes (i.e. WS or NS), display the shapes in descending order!

Student ID : 1234567
Student Name : Tan Ah Meng Elvis

Welcome to Assn2 program!

1) Input sensor data
2) Compute area (for all records)
3) Print shapes report
4) Sort shapes data

Please enter your choice : **4**

a) Sort by area (ascending)
b) Sort by area (descending)
c) Sort by special type and area

Please select sort option ('q' to go main menu) : **b**

Sort by area (largest to smallest) …

Shape [4]
Name : Rectangle
Special Type : NS
Area : 8 units square
Vertices :
Point [0] : (2, 17)
Point [1] : (2, 15)
Point [2] : (6, 15)
Point [3] : (6, 17)

Points on perimeter : (2, 16), (3, 15), (4, 15), (5, 15), (6, 16), (5, 17), (4, 17), (3, 17)

Points within shape : (3, 16), (4, 16), (5, 16)

. . .

Shape [2]
Name : Square
Special Type : WS
Area : 1 units square
Vertices :
Point [0] : (6, 6)
Point [1] : (6, 7)
Point [2] : (7, 7)
Point [3] : (7, 6)

Points on perimeter : none!

Points within shape : none!

APPENDIX E

Submission Instructions (V. IMPT!!)

1) Deliverables

a) All submissions should be in **softcopy**, unless otherwise instructed

b) For the actual files to be submitted, you typically need to include the following:

- word document report (e.g. ***.doc**), save as MS Word 97-2003 format
- the source file(s), (e.g. ***.h**, ***.o**, or ***.cpp** files)
- the executable file, (using Ubuntu g++ compiler), compile into an executable filename with ***.exe** (e.g. csci251_a2.**exe**)

2) How to package

Compress all your assignment files into a single zip file. Please use the following naming format :

<FT/PT>_<Your Grp>_A2_<Stud. No.>_<Name>.zip

Example : FT_TutGrp3_A2_1234567_JohnDoeAnderson.zip

- **<FT/PT>** Use "**FT**" for **F**ull-**T**ime student, "**PT**" if you are **P**art-**T**ime student
- <Your Grp> refers to your SIM tutorial group (e.g. TutGrp1 / TutGrp2 / TutGrp3 / etc.)
- A2 if you are submitting assignment **2**, A3 if submitting assignment **3** etc.
- **<Stud. No.>** refers to your UOW student number (e.g. 1234567)
- **<Name>** refers to your UOW registered name (e.g. JohnDoeAnderson)

3) Where to submit

Please submit your assignment via Moodle eLearning site.

IMPORTANT NOTE :

- To minimize the chances of encountering **UNFORSEEN SITUATIONS** (mentioned below), please do an EARLY SUBMISSION via Moodle.
- Prior to the relevant assignment deadline, you can upload (and replace) your assignment submissions in Moodle **as many times as you like**
- It is your responsibility to confirm that you have submitted the final (and correct version) of your deliverables to Moodle before deadline
- Any submission uploaded to Moodle after deadline will be considered late

In the event of UNFORSEEN SITUATIONS :
(E.g. 3 hrs prior to deadline, there is **proof of** unforseen events like Moodle site down, unable to upload assignment, undersea internet cable damaged by sea urchins, etc)

Please email your single zip file to your tutor at :

csci251@yahoo.com for ***FULL*** TIME students

csci251@yahoo.com for ***PART*** TIME students

In your email **subject** line, type in the following information :

<FT/PT> <Your Grp> <assignment info> <**student number**> and <**name**>

Example:

To : **tutor's email** (see above)

Subject : FT TutGrp3 A2 1234567 JohnDoeAnderson

Note 1 : The timestamp shown on tutor's email Inbox will be used to determine if the assignment is late or not.

Note 2 : After email submission, your mailbox's ***sent folder*** would have a copy (record) of your sent email, please **do not delete** that copy !! It could be used to prove your timely submission, in case the Tutor did not receive your email!

4) When to submit

a) Depending on the time-table, a software demo / testing / presentation for your assignment will be scheduled during the:

- 3rd - 5th lab session for the semester (i.e. lab 3 - 5), for Full Time (FT) students
- 2nd - 4th lab session for the semester (i.e. lab 2 - 4), for Part Time (PT) students

Please consult your tutor for further details. Some time will be allocated for students to demo / present / explain your system design or run test cases during the session.

b) Please refer to the following table on the different deliverables, submission events & deadlines

Assignment #	Submission Deadline (check Moodle for EXACT date-time)		Software Demo / Testing (test cases), during ...	Email Test Case Result files by :
	PT	FT		
1	Lab 2	Lab 3	Lab 2(PT), Lab 3(FT)	End of Lab 2(PT), Lab 3(FT)
2	Lab 3	Lab 4	Lab 3(PT), Lab 4(FT)	End of Lab 3(PT), Lab 4(FT)
3	Lab 4	Lab 5	Lab 4(PT), Lab 5(FT)	End of Lab 4(PT), Lab 5(FT)

Note: (PT) = Part Time Students, (FT) = Full Time Students !

c) IMPORTANT NOTE : Non-submission of any of the above mentioned deliverables will result in ZERO marks! Please check with your Tutor personally if you are unsure!

5) Please help by paying attention to the following …

! VERY IMPORTANT !

PLEASE FOLLOW ALL THE GUIDELINES / REQUIREMENTS IN ALL ASSIGNMENT APPENDICES !!

PLEASE FOLLOW ALL THE SUBMISSION INSTRUCTIONS FROM **1** TO **4** !!

IF YOU ARE **NOT SURE**,

PLEASE **CHECK WITH YOUR TUTOR** DURING LABS / LECTURES !

OR ...

PLEASE EMAIL YOUR ENQUIRIES TO YOUR TUTOR !

MARKS WILL BE DEDUCTED IF YOU FAIL TO FOLLOW INSTRUCTIONS !!

Examples of marks deduction :

- Your deliverables / zip file does not follow naming convention
- You have no email subject / or do not following naming convention
- Your email address / content does not include your name/identity (i.e. tutor cannot easily identify sender)

- Wrong naming or **misleading information** given
  (e.g. putting “A2” in email subject, when you are submitting “A1”)
  (e.g. naming “A1” in your zip file, but inside contains A2 files)

- You email to the WRONG tutor
- Your submission cannot be downloaded and unzipped
- Your program cannot be re-compiled and/or executable file cannot run
- Your report / testing files cannot be opened by Microsoft Word / Excel 2003
- You did not submit / incomplete submission of software demo / testing files
- etc

6) Re-submission administration

After the deadline, (on case-by-case basis), some students / grp may be granted an opportunity for an un-official resubmission by the tutor. If this is so, please adhere to the following instructions carefully:

<Step 1> - Prepare 2 zip files as follows :

Zip up for re-submission, program files according to the following format :

Resubmit_<FT/PT>>_<Your Grp>_A2_<Stud. No.> _<Name>.zip

Example : **Resubmit_FT_TutGrp3_ A2_1234567_JohnDoeAnderson.zip**

Zip up for re-submission, test case results files according to the following format :

Resubmit_<FT/PT>_<Your Grp>_A2_TCResults_<Stud. No.> _<Name>.zip

Example : **Resubmit_FT_TutGrp3_A2_TCResults_1234567_JohnDoeAnderson.zip**

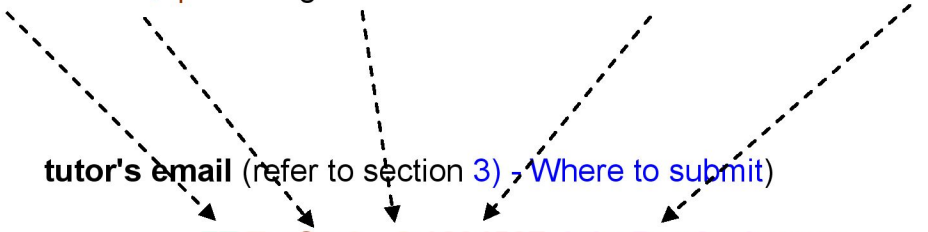
- **<FT/PT>** Use "**FT**" for **F**ull-**T**ime student, "**PT**" if you are **P**art-**T**ime student
- <Your Grp> refers to your SIM tutorial group (e.g. TutGrp1 / TutGrp2 / TutGrp3 / etc.)
- A2 if you are submitting assignment **2**, A3 if submitting assignment **3** etc.
- **<Stud. No.>** refers to your UOW student number (e.g. **1234567**)
- **<Name>** refers to your UOW registered name (e.g. **JohnDoeAnderson**)
- V. IMPT - To prevent Tutor's Inbox from blowing up in his face, each student is only allowed to re-submit ONCE, for each assignment only!

<Step 2>

Please email your 2 zip files to your tutor's email (refer to section 3) - Where to submit)

In your email **subject** line, type in the following information :

Resubmit <FT/PT> <Your Grp> <assignment info> **<student number>** and **<name>**



Example:

To : **tutor's email** (refer to section 3) - Where to submit)

Subject : **Resubmit FT** TutGrp3 A**2 1234567 JohnDoeAnderson**

IMPORTANT NOTE :

Non-submission of any of the above mentioned files, or failure to adhere to submission instructions will result in ZERO marks!

Please check with your Tutor personally if you are unsure!