Student Name: _____ **School:** _____

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

2016 SCHOLARSHIP EXAMINATION

PRACTICAL SECTION

DEPARTMENT	Computer Science
COURSE TITLE	Year 13 Scholarship
TIME ALLOWED	Six hours with a break for lunch at the discretion of the supervisor
NUMBER OF QUESTIONS IN PAPER	Three
NUMBER OF QUESTIONS TO BE ANSWERED	Three
GENERAL INSTRUCTIONS	Candidates are to answer ALL THREE questions. All questions are important. Answer as much of each question as you can. Plan your time to allow a good attempt at each question, but be aware that Question 3 is the most difficult and may take considerably longer than the others.
SPECIAL INSTRUCTIONS	Please hand in listings, notes and answers to written questions, and a Pen Drive or DVD with your program/computer work for each question. In addition please make sure that a copy of each program is printed, or stored as a plain text file. You cannot assume that the examiner has available any special software that might be required to read your files. Candidates may use any text or manual for reference during the examination. Candidates may not have access to the internet during the examination.
CALCULATORS PERMITTED	Yes

TURN OVER

1. **Cheats** (Spreadsheet Use)

In this question you are asked to use a spreadsheet to do calculations and to display the results. We expect that the spreadsheet will be used for all calculations unless the question states otherwise - you will be marked down for performing calculations by hand and directly entering the results. Your work will be graded on three criteria.

(i) The accuracy of your results.

(ii) The skill you show in making use of the capabilities of the spreadsheet.

(iii) The presentation of your results. We have deliberately not provided any instructions concerning layout or formatting and our example graphs lack labels and proper scales.

In this question you are asked to use a spreadsheet to explore and graph some data. The goal is to set up your spreadsheet in such a way as to allow someone else (your customer is someone not very familiar with spreadsheets) to manipulate and draw conclusions from their data. Presenting the information in such a way as to allow this person to work easily is your task. In this question you will work through an imaginary interaction with your customer, trying different approaches to their problem.

The sample data you are to work with is a set of 11 lines, of which the first three will be something like:

```
alpha,60,29,25,63,63,33,36,36,30,71
beta,62,45,36,76,74,41,51,42,41,77
gamma,59,36,35,61,78,51,40,47,39,65
```

The actual data you should use when answering this question is provided for you in a file called Question1.csv.

Your customer is a Mathematics teacher and the data is the test results for the 11 students in their class. The names of the students have been replaced by Greek letter names (for privacy reasons). Each line has the 'name' (Greek letter name) of a student followed by their scores in each of 10 tests. Scores in each test are numbers mostly in the range 0-100, although some students were awarded marks greater than 100 for reasons that are not important in this analysis.

The teacher strongly suspects that two of the students have cheated by sharing answers in all 10 tests.

If the teacher had personally marked the tests, they would probably have noticed cheating. However the marking system used was a little unusual. The papers in each test were distributed amongst the students in another (more senior) class and marked under the teacher's supervision. As a result, no marker marked more than one student's work in any test. No-one was in a position to notice cheating while marking.

Stage A

Create a spreadsheet. Load the 11 lines of data putting each student's data in a separate row with columns for names and test results. *Note point (iii) above. Presentation is up to you.*

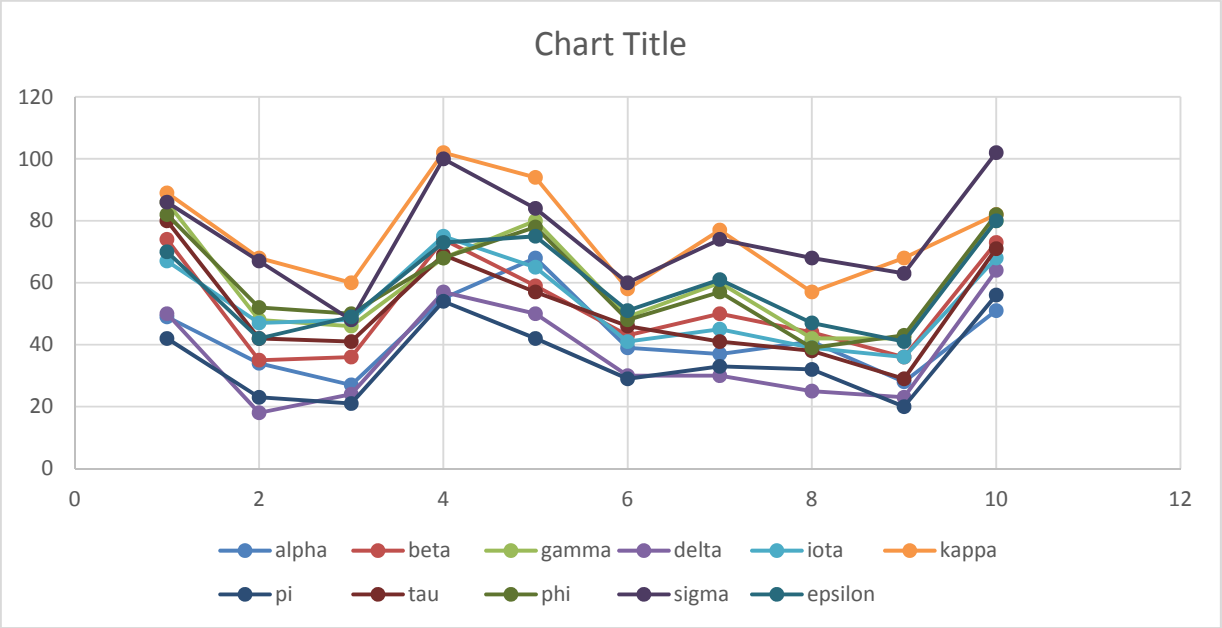
(Question 1 – continued next page)

(Question 1 – continued)

Stage B

The teacher's first idea for finding the cheats is to graph the results and examine the graph.

Add a graph with lines connecting the 10 test results for each student. It might look something like the following (note that the graph printed here uses different data).



The graph proved disappointing. If there were students with identical results, their lines would overlap and would be difficult to see. However, very careful examination of the graph shows that there are no identical lines. It seems that the problem is more difficult than expected. After some thought the teacher realizes that the marking instructions were not precise. Different markers may have given (slightly) different scores for identical work. As a result the cheats will have similar, but not usually identical, grades in each test. The teacher looks at the graph again. It is difficult to find anything. A new form for the graph is needed.

(Question 1 – continued next page)

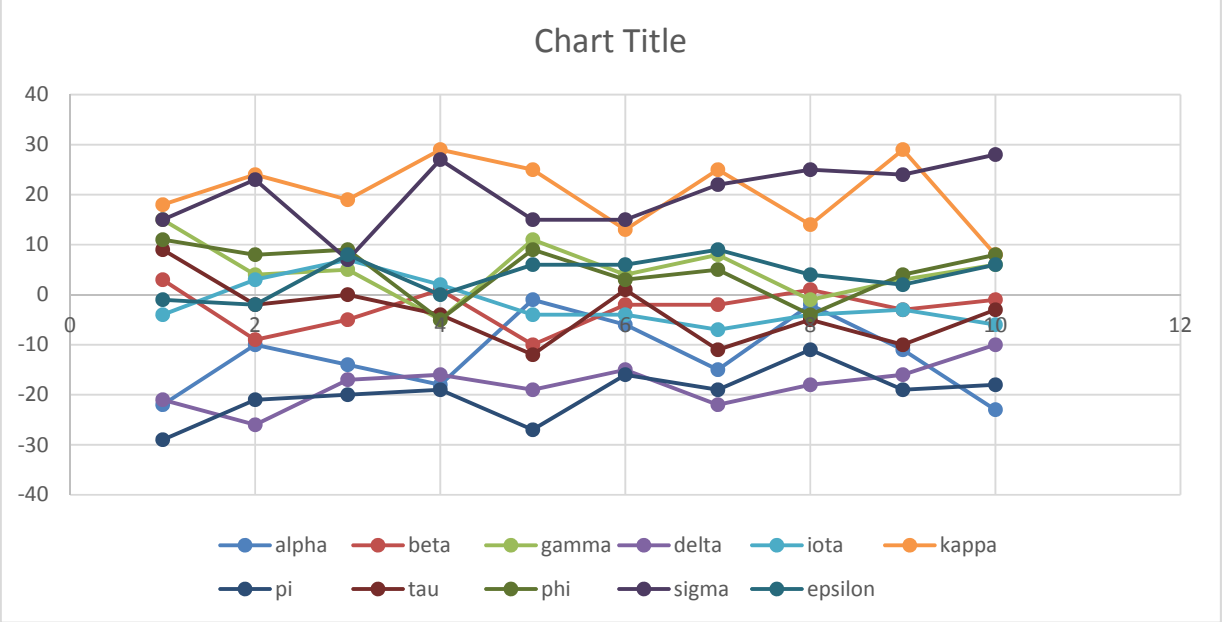
(Question 1 – continued)

Stage C

It is difficult to find similar lines mostly because all the lines go up and down so much – some tests were more difficult than others. You are asked to make a new graph as follows.

For each test, calculate the average mark. Using another part of the spreadsheet, make a new data table showing, for each student on each test, the difference between the student's mark and the class average for that test. For example, if the average mark on the second test was 40, then the new table would show -11 for alpha, +5 for beta and -4 for gamma in the second test column.

Make a graph for your new table. It might look something like this:



This time, the teacher is satisfied. Very carefully looking at the graph shows a pair of lines that are quite similar. In the graph printed above it looks as though 'sigma' and 'epsilon' might be the cheats.

Using your graph, decide which two students are the cheats in Question1.csv.

Stage D

The teacher would like a graph showing only the marks of the two cheating students. Devise a way of allowing the teacher to select two students. Add a new graph to your spreadsheet showing just the test results of the selected students.

(Question 1 – continued next page)

(Question 1 – continued)

Stage E

This part is to be answered on paper. Although you have satisfied your customer, you realize that the power of the spreadsheet allows better analysis of the data. Imagine a new section in your spreadsheet which directly calculates a comparison of student results. It might look like this:

Enter maximum mark difference			5							
	alpha	beta	gamma	delta	iota	kappa	pi	tau	phi	sigma
alpha										
beta	OK									
gamma	OK	OK								
delta	OK	OK	OK							
iota	OK	OK	OK	OK						
kappa	OK	OK	OK	OK	OK					
pi	OK	OK	OK	OK	OK	OK				
tau	OK	OK	OK	OK	OK	OK	OK			
phi	OK	OK	OK	OK	OK	OK	OK	OK		
sigma	OK	OK	OK	OK	OK	OK	OK	OK	OK	
epsilon	OK	OK	OK	OK	OK	OK	OK	OK	OK	SIMILAR

The idea is that the teacher could enter a value into the box at the top (5 has been entered in the screenshot), then the spreadsheet would report as 'SIMILAR' any two students whose marks never differed by more than the given value. The teacher could experiment with the value.

Your task is to work out a way of getting your spreadsheet to do this. Explain your idea with text and diagrams on paper.

2. **Best Score** (Careful and Accurate Programming)

Your programming work in this question will be assessed on two criteria:

(a) Completeness and accuracy of the program.

(b) Good presentation. That is, it should make good use of programming language facilities, be well organised, neatly laid out, and lightly commented.

A new game 'ScholarshipGo' has hit the streets. In this game players explore the world, and catch small creatures.

The game has twelve kinds of creature: Bitsy, Byter, Diskface, Doubler, Driver, Floaty, Floppy, Integer, Linus, Optical, Oss, and Wordfast.

Each creature caught has a strength (a number in the range 0 to 10,000). The goal of the game is to catch as many types of creature as possible; but not just any example of a creature will do. Players must strive to find 'strong' creatures of each type (ie. Creatures with high strength numbers).

Players have 'levels'. When first starting the game, a player is at 'level' 1. As they progress in the game their level can increase. The maximum possible level is 5. To increase in level a player must build a 'strong' collection of creatures and present it for judging. If the collection is complete and the strengths of the creatures are sufficient the player is advanced in level (or completes the game after level 5). If the collection is deemed insufficient, the player loses all progress on that level and must start again.

The game designers have not published the creature strengths required for advancement or winning. This has left many players feeling frustrated, and they turn to you for help. How can they know when a set of creatures has a good chance of being judged to be acceptable.

Leaked information from the game developers establishes some facts. When a player finds a creature, the game decides on its strength. The decision is based on two things: player level and creature type. Higher level players find stronger creatures. In fact there is a secret table in the game specifying the maximum strength to be given for each creature and player level. The numbers in this table are known to be multiples of 200. Strengths given to creatures are a random fraction of the maximum.

You decide to solve the problem by 'crowd sourcing' data. You build a web site on which players can report captures. Each report specifies player level, creature type and creature strength. Your task in this assignment is to write a program which takes a file with collected results from the web site and computes an estimate of the numbers in the table.

You will have access to a file called 'Question2.txt' with data for your program to process. Here are a few lines from the file.

```
5 Floaty 2552
2 Optical 1408
2 Floaty 2262
2 Optical 1518
2 Floaty 1950
5 Floaty 3014
```

(Question 2 – continued next page)

(Question 2 – continued)

There are two reports by level 5 players showing strengths of Floaty's they have found; one is 2552 and the other is 3014. From this we can conclude that the table entry at level 5 for Floaty's is at least 3200 (remember entries are multiples of 200). Similarly the level 2 entry for Floaty's is at least 2400 and the level 2 entry for Optical's is at least 1600.

Your program should read the file and display an estimate of the game's creature strength table.

We suggest that you build this as a 'console' or text interaction program. We do not expect any graphics or elaborate table display. Full marks can be achieved for a 'console' program.

3. **Walking Dogs** (Problem Solving and Programming)

Your programming work in this question will be assessed on two criteria:

(a) Your approach to the problem. We will be looking at your work for evidence that you found good ways of storing the necessary data, and devised algorithms for finding and displaying the requested results. ***Please hand in any notes and diagrams which describe what you are attempting to program, even if you don't have time to code or complete it. You may include comments in your program, or write a description of your program to hand in.***

(b) The extent to which your program works and correctly solves the problem.

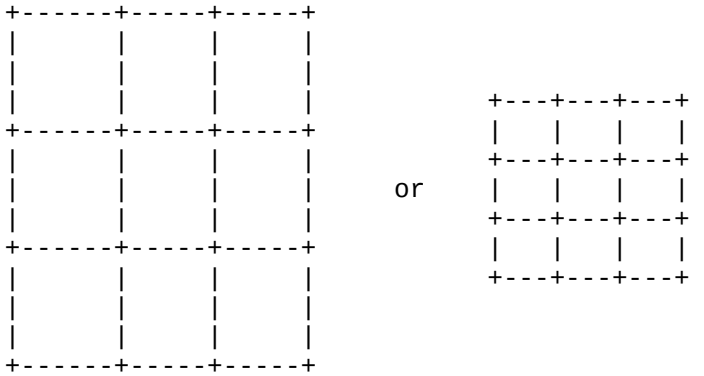
You may find that the programming language you use makes it difficult to produce output as shown in the example implementation steps below. If this is the case, feel free to build your program in a way that suits your circumstances.

Your town has a rather unusual dog exercise area. It consists of an area of large square paving stones, making up a rectangular grid. The entry is at the top left corner and the exit is at the bottom right. Dogs (and their humans) must walk from the entry to the exit. In this problem you are asked to build a program to help dog owners decide on the paths they will use when exercising their dogs.

We will present the problem in stages A to E for you to program. The stages, which are in boxes, are interleaved with explanation of aspects of the problem and some algorithm ideas. We suggest that you build your program in the order given. This will make it likely that you have parts working at the end, even if you don't have time to complete the whole program. However, we also strongly suggest that you read the whole problem statement before starting to program.

Stage A

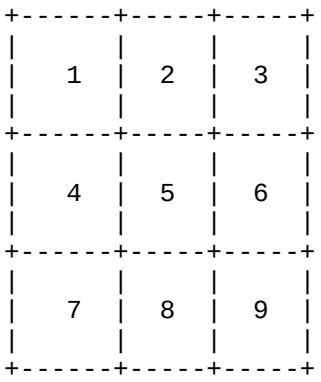
Write a program which prompts the user for the number of rows and number of columns in the grid of paving stones. Your program should display a 'picture' of the grid. For a 3 by 3 grid, your program might output something like the following.



(Question 3 – continued next page)

(Question 3 – continued)

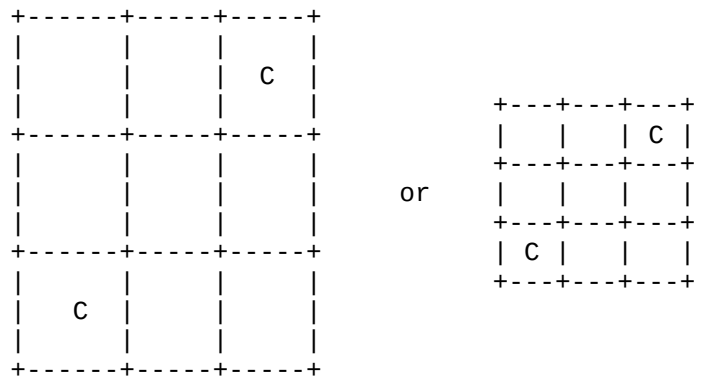
Each paving stone has a flag on each corner, making diagonal movements impossible. The only way to progress from one stone to the next is to cross a side. In the following diagram, the grid squares have been numbered. The flags are at the '+' points. A dog on stone 5 for example can only progress directly to stones 2, 4, 6 and 8. Examples of valid complete paths from start (1) to end (9) are 1, 2, 5, 8, 9 and 1, 2, 3, 6, 5, 8, 9.



Walking a dog on this area has another issue. Neighborhood cats like to sleep on the paving stones. On any given day there may be several cats asleep (no more than one cat can sleep on a paving stone).

Stage B

Extend your program to prompt the user for cat locations. You could accept row and column numbers or you could just number all the stones and accept a stone number. You should then extend your picture to show cats. For example.



The problem now is to find dog walking paths that keep your dog as far away from cats as possible. Of course dogs must never try to walk on a stone where a cat is sleeping, so if there are enough cats or they are unfortunately placed, dog walking may be impossible. To measure distances from dog to cat we will work as follows. The distance from a dog to a cat is the smallest number of edges the dog must cross to reach the cat's square.

(Question 3 – continued next page)

(Question 3 – continued)

Here are some examples where the dogs are one square away from the cat.

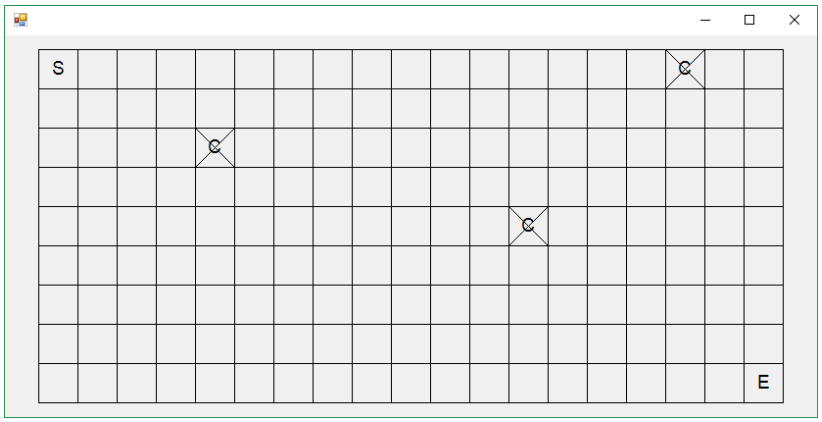
```
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
|   | D | C | D |   |   |   |
+---+---+---+---+---+---+---+
|   |   | D |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
```

Here are some examples where the dogs are two squares away from the cat

```
+---+---+---+---+---+---+---+
|   | D |   |   |   |   |   |
+---+---+---+---+---+---+---+
| D |   | C |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   | D |   |   |   |
+---+---+---+---+---+---+---+
|   |   | D |   |   |   |   |
+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+
```

We can now more precisely state the problem we are trying to solve: find a path from start (top left) to end (bottom right) such that the closest approach to a cat is as distant as possible.

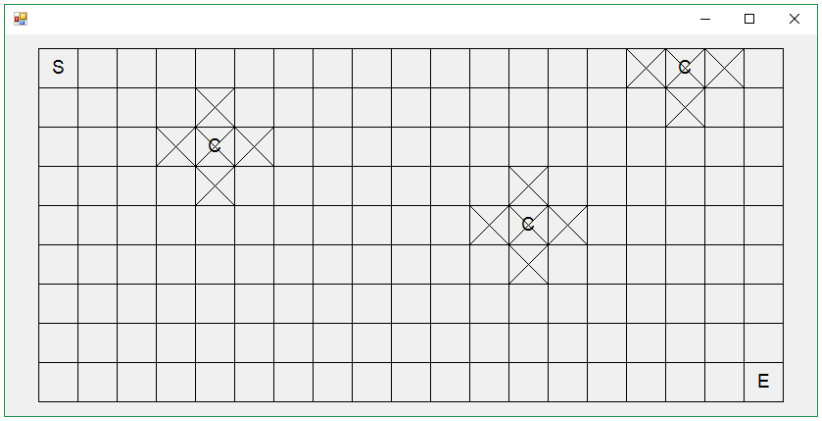
One possible algorithm works as follows: Mark all squares with cats as 'disallowed'. Find out if it is possible to cross the area without stepping on 'disallowed' squares (don't worry about programming this test yet).



(Question 3 – continued next page)

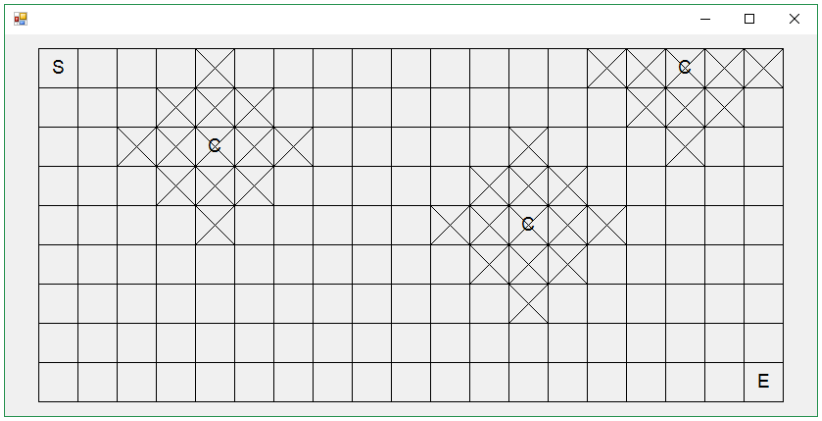
(Question 3 – continued)

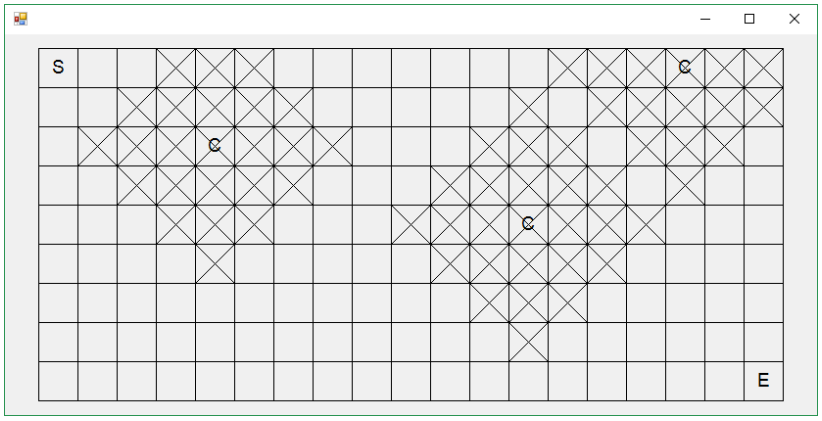
If it is not possible, then there is no way to walk a dog. If it is possible we can ask the question: is it possible to keep further away from the cats. Mark all squares next to cat squares (ie: next to disallowed squares) as also being disallowed. Is it still possible to cross the area?



Any path that avoids disallowed squares at this point is keeping at a distance of at least two from each cat.

Keep going, increasing the disallowed area

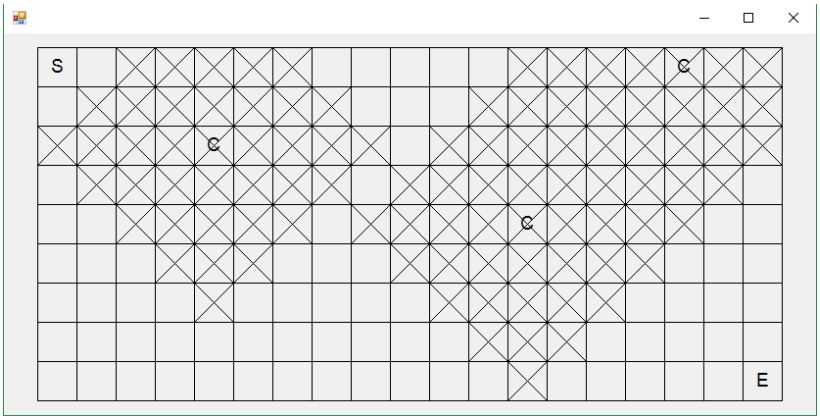




(Question 3 – continued next page)

(Question 3 – continued)

until crossing the area becomes impossible. At this stage you know that a path through the previous version of the area would be a best solution



Stage C

Extend your program to keep track of 'disallowed' squares. Mark all squares with cats as 'disallowed'. Update your display to show disallowed squares.

Stage D

Extend your program to provide a way of growing 'disallowed' areas. This can be done by examining each square to see if it is next to (horizontally or vertically) a 'disallowed' square, and

Stage E

All that is left now it to find an algorithm to decide whether it is possible to cross an area with some squares marked as 'disallowed'. We offer no guidance here. Do your best. If you don't have time to implement an algorithm, write down any ideas you have in as much detail as you can.