THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2020 SCHOLARSHIP EXAMINATION

DEPARTMENT	Computer Science
COURSE TITLE	Year 13 Scholarship
TIME ALLOWED	Five hours with a break for lunch at the discretion of the supervisor
NUMBER OF QUESTIONS IN PAPER	Two
NUMBER OF QUESTIONS TO BE ANSWERED	Two
GENERAL INSTRUCTIONS	Candidates are to answer BOTH questions. Each question is important. Answer as much of each question as you can. Note that Question 2 is significantly more difficult than Question 1. Plan your time to allow a good attempt at each.
SPECIAL INSTRUCTIONS	Please hand in notes and answers to written parts of the 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 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 except for access to online documentation for the software they are using.
CALCULATORS PERMITTED	Yes

1. **Scores** (Careful and Accurate Programming)

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

(a) Completeness and accuracy of the program. It may be that this problem statement does not state exactly what the program should do under all circumstances. If you find a situation of that nature, choose a solution and write down, either on paper or in the comments of your program what the difficulty was and how you chose to resolve it.

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

(c) Careful checking. Wherever possible check input from the program user in case they have made errors.

In this question you are asked to write a program to process results from the New Zealand programming contest. The NZ Programming Contest works as follows.

- ⇨ There is a set of 16 problems identified by letters of the alphabet from A to P.
- ⇨ Teams of three people work together to solve problems.
- ⇨ When a team thinks it has a solution to a problem, they submit their program for judging.
- ⇨ A program is judged to be correct or not.
- ⇨ Teams are awarded points for each problem correctly solved.
- ⇨ Problems A, B, C and D are considered easy to solve and are worth 3 points each.
- ⇨ Problems E, F, G and H are worth 10 points each.
- ⇨ Problems I, J, K and L give 30 points.
- ⇨ The most difficult problems M, N, O and P are each worth 100 points.
- ⇨ The winning team is the one with the most points at the end of the contest.
- ⇨ The result can be a tie between two or more teams.
- ⇨ In the case of a tie for highest score, all teams with that score are declared joint winners.

Your task is to write a program which interacts with a user allowing that user to enter the names of the teams and the results from each solution attempt judged (team number, problem letter, and result). It should report the names of winning teams.

The transcript of a sample interaction with such a program is given on the next page. In the transcript, information entered by the user is shown in **bold** type. You don't have to follow this style of data entry or format results in the same way. The sample is just here to show the kind of interaction expected of your program.

QUESTION 1 CONTINUES ON THE NEXT PAGE

```
NZPC Results Analysis

How many teams are competing: 5
  Enter name of team 1: Code Raiders
  Enter name of team 2: Sleepless
  Enter name of team 3: Recursive
  Enter name of team 4: Extremes
  Enter name of team 5: Algorithmics

Enter Results
  Is there another result (y/n): y
    Team number: 1
    Problem: A
    Solved? (y/n): n
  Is there another result (y/n): y
    Team number: 2
    Problem: N
    Solved? (y/n): n
  Is there another result (y/n): y
    Team number: 2
    Problem: I
    Solved? (y/n): y
  Is there another result (y/n): y
    Team number: 2
    Problem: J
    Solved? (y/n): y
  Is there another result (y/n): y
    Team number: 1
    Problem: P
    Solved? (y/n): y
  Is there another result (y/n): n

The winning team was "Code Raiders" with 100 points
```

2. **Cars** (Problem Solving and Programming)

Your programming work in this question will be assessed on three 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.

(c) The extent to which you use results from your program to explore the problem presented.

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.

The first steps in developing New Zealand's network of roads were the building of individual connections between towns. As the years passed, more and more connections were built, until we had a system that allowed people to drive from most starting points to most possible destinations within each of our islands. Some of those connections, joined end to end, came to be thought of as our state highways. State Highway 1 was formed as a sequence of town to town connections all the way from the far North of the North Island to the bottom of the South Island (with ferries providing the link between islands). In the early years of motoring, taking a trip in sections, visiting town after town, must have seemed sensible. Having the roads pass through the central business districts of each town allowed travellers to stop and provided opportunities for businesses in each town. For safety though, those sections through towns had to have reduced speed limits. Now, with high traffic volumes, the slow sections in each town constitute major bottlenecks for traffic flow. One by one, bypass roads are being put in place so that traffic can continue at full speed past each town.

The (fictitious) town of Foxvin, in the lower half of the North Island is facing the bypass issue. The town businesses don't want to lose the patronage of travellers; but drivers don't want to be delayed in the traffic jams that build up on either side of town during busy times. The town council has an idea which they hope can satisfy both groups of people. At present the North and South connecting roads into the town have 100 km/hour speed limits. The town road segment connecting them has a 50 km/hour limit (as shown in Diagram A on the next page). The council's idea is to add a second 50 km/hour road segment, giving drivers a choice of roads within the town. The new segment will also pass through the central business district, so that all drivers will pass businesses in the town and have the opportunity to stop (Diagram B). The council hopes that two 50 km/hour roads will carry all the traffic from the 100 km/hour roads without having cars stopped in traffic jams at the borders of the town.

QUESTION 2 CONTINUES ON THE NEXT PAGE

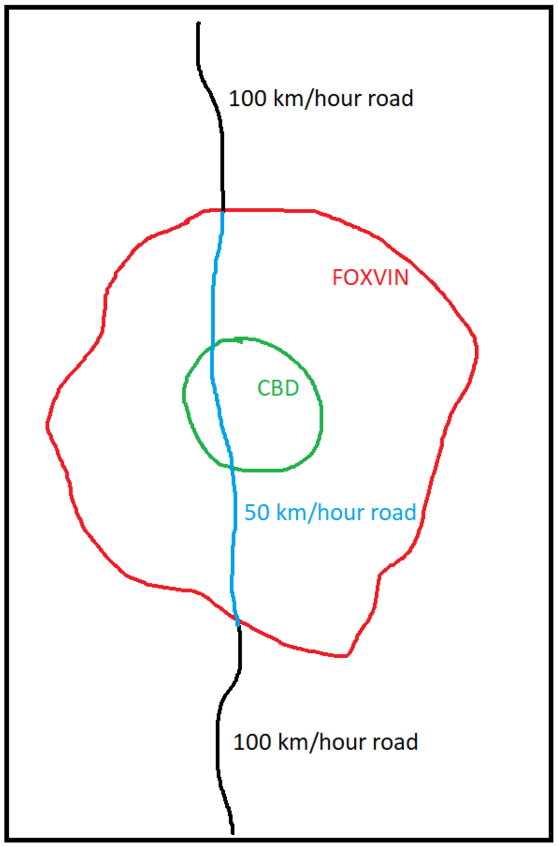

Diagram A

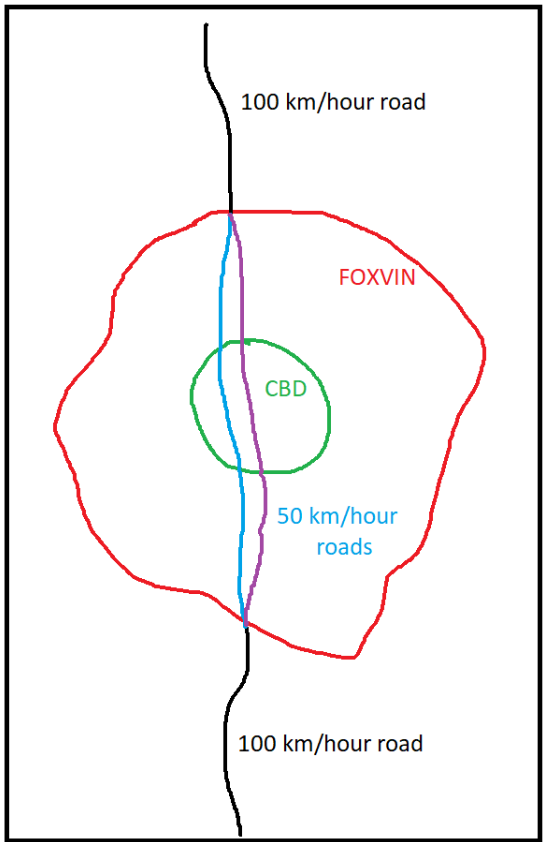

Diagram B

The Foxvin town council has called on you to write a traffic simulation program to determine whether their solution could work. This question presents the problem in stages for you to program. The stages include further explanation of the expected behavior of drivers that will be required as part of the simulation and recommendations for building the simulation itself. 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 through all the stages before starting to program.

As you complete each stage, you should save working versions of your program. These will help us see what you have achieved, especially if you have difficulties in later stages. Instructions will be given in some detail for the first stages. Later stages require that you develop the code yourself.

Stage A: Making a road.

For the purposes of simulation we will make some simplifying assumptions. First, the shape of the road isn't important. We can treat it as a straight line. Second, North to South traffic will behave in a very similar way to South to North traffic. We will only deal with traffic flowing in one direction. Therefore, we are going to simulate traffic flowing in one direction along a straight road.

When developing simulations, it is very helpful to have some graphics – pictures showing us what is happening. As a first step we will 'draw' a road on the screen. Because this examination is intended to be independent of the programming language you use, we

QUESTION 2 CONTINUES ON THE NEXT PAGE

cannot assume that you have any advanced graphics capabilities. Instead we will fall back on using text to make pictures. Because screens are wider than they are high, we will make our road a straight line from left to right (and our cars will travel left to right) Draw a road like this; using '+' characters like a fence on the sides of the road, and spaces through the middle as the road surface. The sample has a road 100 characters long.

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Write a program which can display a road as shown.

Stage B: Different speeds.

To simulate traffic flowing through a town, we need the first(left) and last(right) parts of the road to be at 100 km/hour and the middle part (through the town) to be at 50 km/hour. To show the 50 km/hour part we can change the characters used on the sides. In the text following we will call the middle part the 'slow' part.

We need to make some choices about scale for our graphics.

The simulation doesn't need to have very long pieces of road. The example given below has been set up to represent a road length of ¼ km (250 metres) with the slow part starting 0.1 km (100 metres) km from the left and being 0.08 km (80 metres) long. You may want to make your simulated road longer or shorter when you are doing experiments, but we suggest that you start with the given values.

Draw the road with fast and slow parts.

```
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
```

Stage C: The first car.

Before drawing a car we have to decide how long it will be. The lengths of most ordinary cars on New Zealand roads are in the range of 2.7 metres to 5 metres. For the purposes of this simulation we will take the larger value and assume that all cars on our road are 5m long. With the scales above this translates to a length of 2 characters, so we can represent one car as 'CC'. Note that the displayed size of a car will change if you alter the length of your displayed road or of your simulated road, so your program should calculate the size (number of characters).

Draw a car at some distance (d) along the road. Here is an example of a car 50 metres from the start of the road. (The back end of the car is 50 metres from the start.)

```
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
                    CC
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
```

QUESTION 2 CONTINUES ON THE NEXT PAGE

Stage D: Moving the car.

Using text graphics, animation isn't easy. Instead we will draw the road repeatedly to show what it looks like at different times.

Simulation can take advantage of the rapid calculation capability of a modern computer in order to get results with repeated simple calculations. For this simulation we will calculate positions and speeds of cars 10 times per second. At each time point, for each car, we will set a speed and assume that the car continues at that speed for the next 1/10th second. It is not necessary to do acceleration calculations. We can just assume that the car instantly changes speed when needed. So, every 1/10th second, for each car:

```
Decide on a preferred speed ($v$ in km / hour)
Calculate the distance the car could travel in 1/10 sec
        This will be $v * \frac{1}{10}$ / 3600 km (because there are 3600 seconds in 1 hour)
Update the position of the car
```

Draw a picture every 1/10 second, showing a car moving at 100 km/hour. Here are the results from half a second of simulation. Notice that the car sometimes moves along two characters and sometimes one character. That is ok – the graphics isn't very precise, but if we store the position using a float, and only round to a character position for display, the final result will be accurate enough.

```
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
CC
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
  CC
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
   CC
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
    CC
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
     CC
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
      CC
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
```

Stage E: Display occasionally.

It is going to be impractical to display results every 1/10 of a second if the simulation is going to run for many minutes. Change your program so that the calculations continue to be done every 1/10 second, but results are only displayed once per second.

QUESTION 2 CONTINUES ON THE NEXT PAGE

Stage F: Changing speed.

Change your program so that the car slows down on the slow portion of the road. As a first experiment make the slow speed 50 km/hour. You should be able to clearly see the effect in your program output. Try different speeds.

Stage G: Reporting speed

It can be difficult to see how fast a car is moving by looking at text pictures. One way of roughly reporting speed is to change the appearance of the car depending on its speed. In the following image the car is shown with the 'F' character travelling at 100 km/h and 'S' when travelling at 50 km/h. Later you may find it helpful to show other speeds or speed ranges.

```
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
                                      FF
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
                                         SS
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
                                          SS
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
```

Change your program to vary its output with car speed.

Stage H: More than one car.

Traffic jams are only a problem when there is a lot of traffic. Our simulation will assume that the incoming road is 'full' – i.e. carrying as many cars as it possibly can. So, how many cars can fit on the road? This is determined first by the lengths of the cars. The most full a road can be is a bumper to bumper traffic jam. When cars are moving however, they must be spaced further apart. The road code has clear rules. For the purposes of this simulation we will take the rule to be that cars must keep a gap of 1 car's length for every 10 km/h of speed. At 100 km/h this is a gap of 10 cars lengths (v / 10 = 100 / 10) from the front of a car to the back of the car in front. At 50 km/h the gap is 5 car lengths.

We can modify the algorithm for deciding how a car moves on each calculation step to:

```
Decide on a preferred speed (v in km / hour)
Calculate the distance the car could travel in 1/10 sec (d km)
⇨ Reduce the distance if necessary to avoid coming too close to the car in front (reduce d)
    Note that we can now calculate the actual speed as $d * 3600 / \frac{1}{10}$
Update the position of the car
```

Update your simulation to start cars at the left hand side as rapidly as is possible, and have the cars obey the following distance rule given.

QUESTION 2 CONTINUES ON THE NEXT PAGE

Stage I: What happens?

This stage requires a written answer. What does your simulation do? Do you get a traffic jam? Discuss. Your written answer can include screen dumps from your simulation.

Stage J: What drivers really do.

Note that this section asks you to implement driving behaviour which is not legal on the roads in New Zealand. DO NOT DO THIS WHEN YOU ARE DRIVING A REAL CAR.

The Foxvin council has observed something they call Impatient Driver behaviour. If a driver has a speed limit of 100 km/h and is following behind a car travelling slower than 95 km/h they will keep driving at 100 km/h until the gap between the two cars is reduced to ½ car length. In other words they quickly pull up and follow very closely behind slower drivers. This is in fact a general behaviour: a driver(A) behind another car travelling at less than 95% of A's legal speed limit will continue at A's limit until they are ½ a car's length behind the other car and follow in that position.

Change your program to make all cars implement impatient driver behaviour.

Stage K: What happens with impatient drivers?

This stage requires a written answer. What does your simulation do now? Try different 'slow' speeds. Under what circumstances do you observe a traffic jam? Discuss.

Stage L: Two slow roads.

Finally we reach the point of testing the Foxvin council's idea. What if there were two slow roads through town. In the simulation we can make the equivalent of a second road by widening our road at the slow point. Here is an image with road drawn double width and '*' characters in all the places cars can drive.

Note: We could make this look better by moving the fences on the 100 km/h sections, but it doesn't improve the clarity of the graphics when there are cars on the road.

```
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
****************************************************************************************************
                                        ********************************
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
```

Here it is without the '*' characters

```
++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++################################++++++++++++++++++++++++++++
```

Change your program to draw a double width road.

QUESTION 2 CONTINUES ON THE NEXT PAGE

Stage M: Using two slow roads.

Change your program so that cars make use of the double slow road. You should have each car make a choice and then stick with it. If the doubling were implemented as a two lane road, cars could move back and forth between the lanes. However, the real proposal for Foxvin has separate roads and each driver will have to choose one and follow it. You will have to decide how cars select the slow road that they are going to use.

Stage N: What happens?

This stage requires a written answer. What does your simulation do now? Does the Foxvin council's idea work to keep traffic flowing? Discuss. Your written answer should include screen dumps from your simulation.

Stage O: Further work.

The simulation, or a modified version of it, can be used to explore other options. Councillors would like to know if the speed limit in the town could be further reduced (below 50 km/h), and if so, to what extent. There is also interest in adding a third slow road. What difference would that make? Explore each of these options and report!