VIRTUAL SCHOOL VICTORIA

2021

SOLUTIONS

ALGORITHMICS UNIT 3 & 4

Trial Exam 1: 2021

Reading Time: 15 minutes
Writing time: 120 minutes (2 hours)

QUESTION AND ANSWER BOOK

Section	*Number of questions*	*Number of questions to be answered*	*Number of marks*
A	20	20	20
B	7	7	80

- Students are permitted to bring into the examination room: pens, pencils, highlighters, erasers, sharpeners, rulers and one scientific calculator.
- Students are NOT permitted to bring into the examination room: blank sheets of paper and/or correction fluid/tape

Materials supplied

- Question and answer book of 24 pages
- Answer sheet for multiple-choice questions

Instructions

- Write your student number in the space provided above on this page.
- Check that your name and student number as printed on your answer sheet for multiple-choice questions are correct, and sign you name in the space provided to verify this.

- All written responses must be in English, point form is preferred.

Students are NOT permitted to bring mobile phones and/or any other unauthorised electronic devices into the test room.

IMPORTANT NOTE:
The VCAA Exam will include the Master Theorem in this form.

Use the Master Theorem to solve recurrence relations of the form shown below.

$$T(n) = \begin{cases} aT\left(\dfrac{n}{b}\right) + kn^c & \text{if } n > 1 \\ d & \text{if } n = 1 \end{cases} \qquad \text{where } a > 0, b > 1, c \geq 0, d \geq 0, k > 0$$

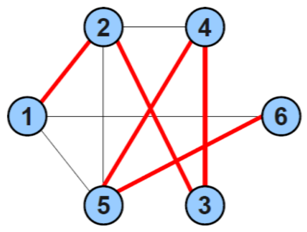
$$\text{and its solution } T(n) = \begin{cases} O(n^c) & \text{if } \log_b a < c \\ O(n^c \log n) & \text{if } \log_b a = c \\ O(n^{\log_b a}) & \text{if } \log_b a > c \end{cases}$$

The VCAA form of Master Theorem is equivalent to the form of Master Theorem taught in our class by consideration of log laws. $\log_b a = c \Leftrightarrow a = b^c \Leftrightarrow \frac{a}{b^c} = 1$ $\log_b a < c \Leftrightarrow a < b^c \Leftrightarrow \frac{a}{b^c} < 1$ $\log_b a > c \Leftrightarrow a > b^c \Leftrightarrow \frac{a}{b^c} > 1$	$T(n) = aT\left(\frac{n}{b}\right) + f(n^k)$ • $\frac{a}{b^k} < 1$ then $O(n^k)$ • $\frac{a}{b^k} = 1$ then $O(n^k \log_b n)$ • $\frac{a}{b^k} > 1$ then $O(n^{\log_b a})$

SECTION A – Multiple Choice – select one option only

Question 1

Consider the following problem:
For a graph G of nodes and edges, is there a simple path that goes through every node exactly once?

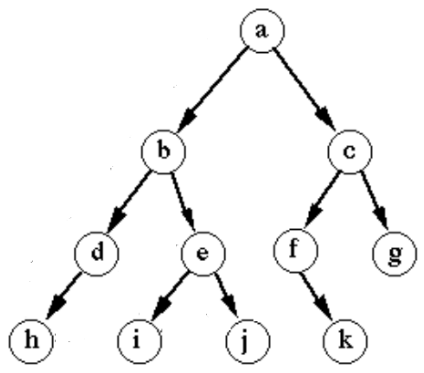


Which description best fits this problem.

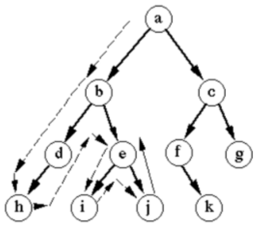
A. Feasible, P-class **B.** Uncomputable, NP-Complete **C.** Undecidable, NP-Hard **D.** Intractable, NP	Answer D Hamiltonian path solution cannot be found in polynomial time, so it's intractable, but should a solution be found it can be verified in polynomial time, making this problem NP

Question 2

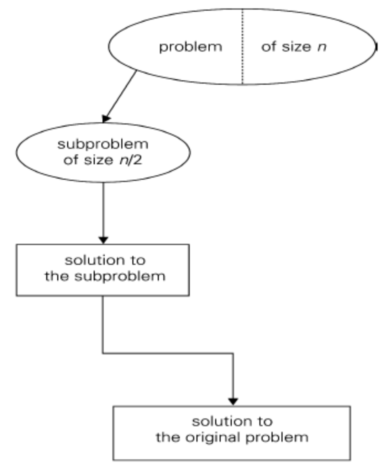
If the following graph is traversed using Depth First Search starting at node a, and using alphabetic order where multiple options exist.



The order of nodes processed will be:

A. a b d e h i j c f k g **B.** a b c d e f g h i j k **C.** a b e i j d h c g f k **D.** a b d h e i j c f k g	**Solution** Option D  a b d h e i j c f k g

Question 3

A recursive algorithm has the property of one recursive call which decreases the size of the problem by 0.5 until the base case is reached, where a constant amount of work is done. Outside of the recursion a linear amount of work is done relative the input size. The **recurrence relation** representing the number of actions with respect to the input size of n would be:	 problem of size *n* subproblem of size *n*/2 solution to the subproblem solution to the original problem

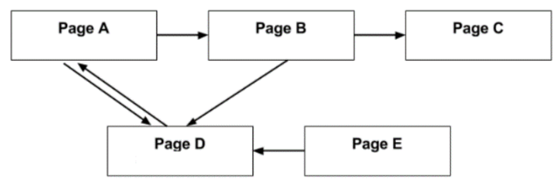
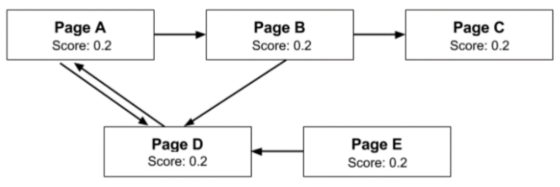
A. T(n)=T(n/2)+n, T(1)=O(1) **B.** T(n)=T(n/2) + O(1), T(1)=O(1) **C.** T(n)=O(nlogn) **D.** T(n)=T(n-1)+n, T(1)=O(1)	**Answer is A** **One recursive call, data halved**

Question 4

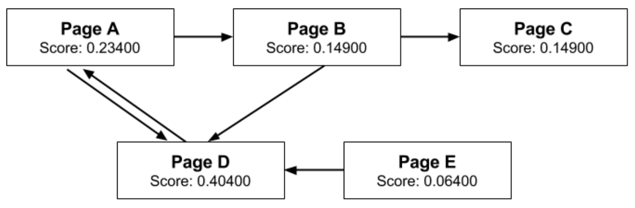
If all edges have the same weight in an undirected graph, which algorithm will find the shortest path between two nodes more efficiently?

A. Dijkstra **B.** Bellman-Ford **C.** Depth-First Search **D.** Breadth-First Search	Answer : D Breadth-First Search has time complexity O(|V| + |E|).

Question 5

The Page rank algorithm is run on the graph shown below	After initialisation the Page rank scores for each page are shown below
	 Page A Score: 0.2 Page B Score: 0.2 Page C Score: 0.2 Page D Score: 0.2 Page E Score: 0.2

After iteration 1 the Page rank score for Page D is:**.**

A. 0.40400 **B.** 0.06400 **C.** 0.14900 **D.** 0.23400	The Answer is A  Page A Score: 0.23400 Page B Score: 0.14900 Page C Score: 0.14900 Page D Score: 0.40400 Page E Score: 0.06400

Question 6

Consider the following problem:

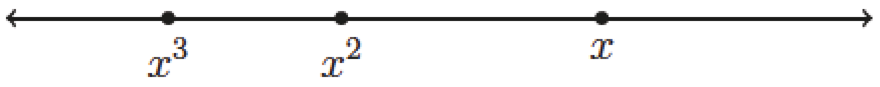
Given a graph G and an integer k, is there a spanning tree whose edges sum to less than k?

What is the classification of the problem above?

A. Decision problem **B.** Minimum Spanning Tree problem **C.** NP problem **D.** Hard problem	The answer to a decision problem is yes/no. The search for a spanning tree might have multiple correct answers, e.g., a given graph might have multiple minimum spanning trees. These can be checked against the value of k either <k or not. Answer is A

Question 7

If x, x^2 and x^3 lie on a number line in the order shown below, which of the following could be the value of x?



A. -2 **B.** $-\frac{1}{2}$ **C.** $\frac{3}{4}$ **D.** 1	Answer is C We have $0 < x^2 < x$ so that x is positive and $x < 1$. The only possibility is $x = \frac{3}{4}$, and $x^3 = \frac{27}{64}$, $x^2 = \frac{9}{16} = \frac{36}{64}$ and $x = \frac{3}{4} = \frac{48}{64}$, hence (C).

Question 8

Which of the following best describe simulated annealing used as a heuristic?

A. Returns an approximation to the optimal solution using a cooling schedule of randomisation. **B.** Returns an approximation to the optimal solution where there is no randomisation. **C.** It will not return an optimal solution when there is a cooling schedule of randomisation. **D.** Uses purely greedy methods for an approximation to the optimal solution.	Answer is A

Question 9

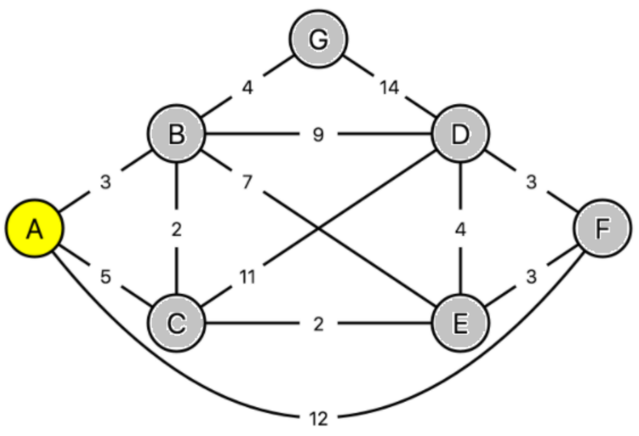
The missing terms in order for the Priority Queue signature shown below are:

$$\begin{array}{ll} \textit{name priorityQueue}; & \\ \textit{ops} & \textit{newPriorityQueue} \ : \ \rightarrow \textit{priorityQueue}; \\ & \textit{enqueue} : \textit{priorityQueue} \times \textit{element} \times \boxed{} \rightarrow \textit{priorityQueue}; \\ & \textit{minElement} \ : \textit{priorityQueue} \rightarrow \textit{element}; \\ & \textit{dequeueMin} : \textit{priorityQueue} \rightarrow \boxed{}; \\ & \textit{isEmpty} \ : \textit{priorityQueue} \ \rightarrow \boxed{}; \end{array}$$

A. item, item, Boolean **B.** rank, item, priorityQueue **C.** rank, priorityQueue, Boolean **D.** rank, element, element	**Answer C** *name priorityQueue*; *ops* *newPriorityQueue* : → *priorityQueue*; *enqueue* : *priorityQueue* × *element* × *rank* → *priorityQueue*; *minElement* : *priorityQueue* → *element*; *dequeueMin* : *priorityQueue* → *priorityQueue*; *isEmpty* : *priorityQueue* → *boolean*;

Question 10

Using the graph below,



if we apply Dijkstra's algorithm to find the shortest distance between node A and all the others, in what order do the nodes get included into the visited set (i.e their distances have been finalized)?

A. B C F G E D **B.** B C G E F D **C.** C B E F G D **D.** C B E G F D	Answer : B The nodes are added to the visited set in terms of the shortest distance to the source.

Question 11

Which of the following is NOT an example of an NP problem?

A. 3 colouring of a given graph **B.** Travelling salesman problem **C.** Knapsack **D.** Halting Problem	Answer is D: Halting Problem is outside NP

Question 12

What is Searle's "Chinese room" thought experiment supposed to show?

A. That computers aren't yet able to simulate the human ability to understand **B.** That understanding involves more than the ability to formally reproduce appropriate outputs **C.** That it is only possible for systems to demonstrate understanding **D.** That no machine can demonstrate genuine understanding	Answer is B

Question 13

Assuming $P \neq NP$, which of the following is true?

A. NP-Complete = NP **B.** NP-Complete ∩ P = ψ {empty set} **C.** NP-Hard = NP **D.** P = NP-Complete	Answer is B

Question 14

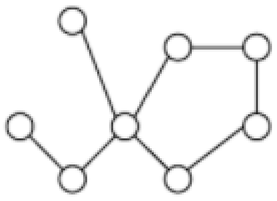
The following recurrence relation $T(n) = 2^n\, T(n/2) + n^n$ gives information about how much work is done by a particular recursive algorithm.

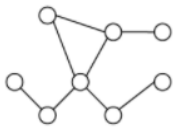
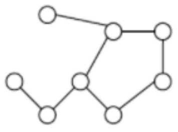
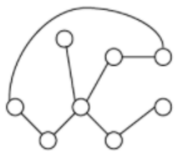
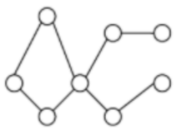
Using the Master Theorem it can be represented by the following function of the input size n.

A. O(nlogn) **B.** O(2^n) **C.** O(n!) **D.** Master Theorem does not apply as the number of recursive calls made is not a constant value	Answer D: Master Theorem does not apply since a is not a constant

Question 15

Which graph below is isomorphic to this graph:



A 	**B** 	**C** 	**D** 	**Answer C**

Question 16

What does the following mystery function defined in pseudocode below do?

```
Function mystery(a, b)
// Input a an integer
// Input b an integer

    if (b == 0) then
       return 0
    else
       return (a + mystery(a, b-1))
    end if
end function
```

A. Returns $a + b$

B. Returns $a + ab$

C. Returns ab

D. Returns a^b

Answer is C

Explanation: The function adds x to itself y times which is x*y.

Question 17

Consider a situation where you have to write a function to calculate a number raised to a power such as x^n where x can be any number and n is a positive integer. What can be the best possible time complexity of your power function using an advanced design pattern?

A. O(n)

B. O(logn)

C. O(log(logn))

D. O(nlogn)

Answer B

Using Divide & Conquer and the properties

power(x, n) = power(x, n / 2) × power(x, n / 2); // otherwise, n is even

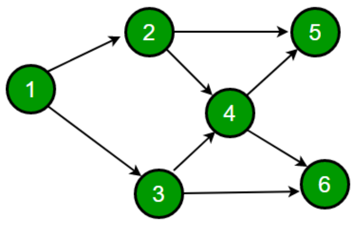
power(x, n) = x × power(x, n / 2) × power(x, n / 2); // if n is odd

```
function power(x, n):

    # base condition
    if n == 0 then
        return 1
    else

       # calculate subproblem recursively
       pow = power(x, n / 2)
       if n is odd then
           return x * pow * pow
       else
           return pow*pow
       end if
    end if
end function
```

https://www.techiedelight.com/power-function-implementation-recursive-iterative/

Question 18

Consider the Directed Graph shown



Which of the following is NOT a topological ordering?

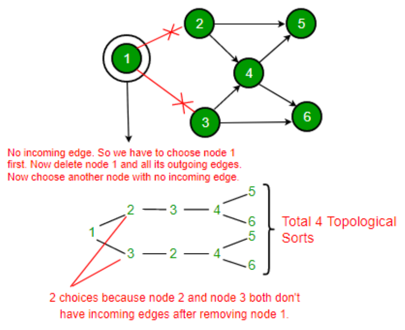
A. 1 2 3 4 5 6

B. 1 3 2 4 5 6

C. 3 2 4 1 6 5

D. 1 3 2 4 6 5

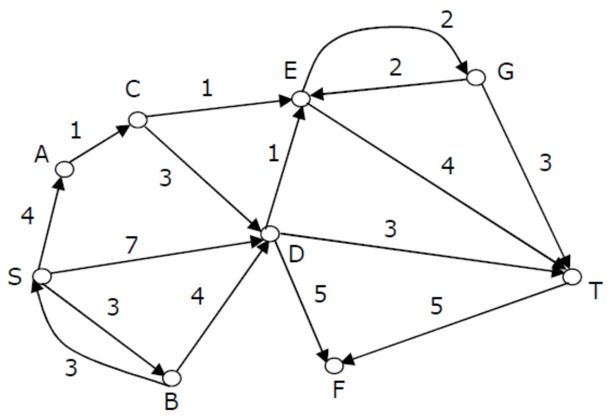
Answer C



No incoming edge. So we have to choose node 1 first. Now delete node 1 and all its outgoing edges. Now choose another node with no incoming edge.
Total 4 Topological Sorts
2 choices because node 2 and node 3 both don't have incoming edges after removing node 1.

Question 19

Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Which one will be reported by Dijkstra?s shortest path algorithm? Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.



A. SACET

B. SDT

C. SBDT

D. SACDT

Answer A
When the algorithm reaches vertex ‘C’, the distance values of ‘D’ and ‘E’ would be 7 and 6 respectively. So the next picked vertex would be ‘E’

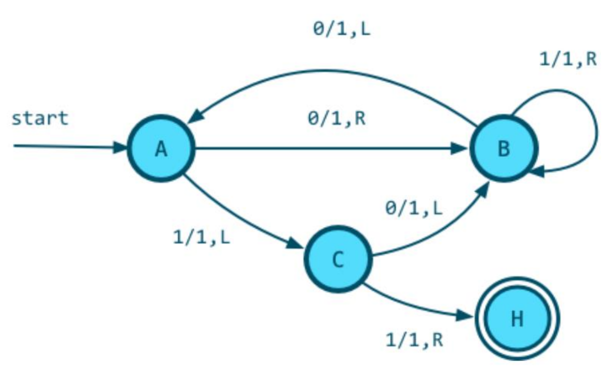
Question 20

Consider the following three functions: $$f1(n) = 10^n$$ $$f2(n) = n^{logn}$$ $$f3(n) = n^{\sqrt{n}}$$ Which one of the following options arranges the functions in the increasing order of asymptotic growth rate? **A.** $f3, f2, f1$ **B.** $f2, f1, f3$ **C.** $f1, f2, f3$ **D.** $f2, f3, f1$	**Answer D** **Explanation:** On comparing power of these given functions : f1 has n in power. f2 has logn in power. f3 has √n in power. Hence, f2, f3, f1 is in increasing order. Note that you can take the log of each function then compare.
https://www.geeksforgeeks.org/gate-gate-cs-2021-set-1-question-13/	

SECTION B – Extended Response Questions Answer all questions in the space provided.

Question 1 (14 marks)

The following Turing Machine (TM) has been as a directed graph, with states represented by nodes and transitions labelled on the edges.



Edge transition description:	TM Symbols: {1,0}
in/*out*,R *in*/*out*,L	*in* is the input symbol *out* is the output symbol R moves one symbol to the right L move one symbol to the left

a) For the above Turing Machine describe the steps taken to the output when it is given the following input tape, with the current state and position on the input tape shown. (3 marks)

A
↓

	0	1	1	0	0	1

1	Starts in state A looking at symbol 0, overwrite with 1 and move right and new state B B ↓ $\boxed{}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}$	4	In state B looking at symbol 0, overwrite with 1 and move left and new state A A ↓ $\boxed{}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{1}$
2	In state B looking at symbol 1, overwrite with 1 and move right and new state B B ↓ $\boxed{}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}$	5	In state A looking at symbol 1, overwrite with 1 and move left and new state C C ↓ $\boxed{}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{1}$
3	In state B looking at symbol 1, overwrite with 1 and move right and new state B B ↓ $\boxed{}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{1}$	6	In state C looking at symbol 1, overwrite with 1 and move right and new state H H ↓ $\boxed{}\boxed{1}\boxed{1}\boxed{1}\boxed{1}\boxed{0}\boxed{1}$

There are no further transitions defined for state H so we halt.

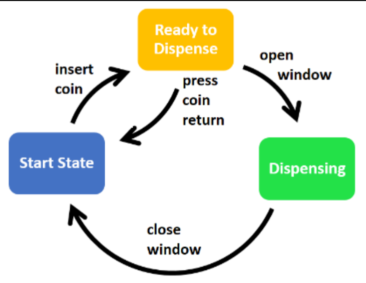
b) Convert the Turing Machine shown above into a table of instructions. (3 marks)

Current State	Scanned Symbol	Print Action	Move Action	Next State
A	**0**	**1**	**R**	**B**
A	**1**	**1**	**L**	**C**
B	**0**	**1**	**L**	**A**
B	**1**	**1**	**R**	**B**
C	**0**	**1**	**L**	**B**
C	**1**	**1**	**R**	**H**
H				

Question 1 (continued)

Finite State Automata (FSA) are very simple machines, an example is a simple vending machine that will dispense a candy bar when a dollar coin has been inserted.

- There are four actions possible: insert a coin, press `coin return', open the window to take the candy bar, and close the window again.
- There are three states: the start state, ready to dispense state, dispensing state.
- The FSA is shown in the diagram at the right.



c) With consideration of the definition of a FSA provided above, outline the similarities and the differences between a FSA and a TM. (3 marks)
Both have defined states and defined transitions.
Once a FSA has "seen" a symbol it is lost; in the TM the symbol can still be retained on the tape. The FSA has no "memory" except that which is implied in the choice of states; in the TM there is an unbounded memory on the tape.

d) Formally what are the definition of problems in terms of solvability and verifiability in **class P** and in **class NP** with respect to Turing machines? (3 marks)
The complexity class P is defined as the set of decision problems that can be solved and verified by a deterministic Turing machine in polynomial time
NP is the class of problems that are solvable by a nondeterministic Turing machine in polynomial time, whose solutions are verifiable by a deterministic Turing machine in polynomial time.

Consider this lock that requires a combination to be opened.

e) Can this lock be opened by a randomly chosen student? Using Computer Science conventions describe what kind of problem this is and discuss which complexity class this problem belongs to with justifications of your selection. (2 marks)
NP is the class of problems that are solvable by a nondeterministic Turing machine in polynomial time, whose solutions are verifiable by a deterministic Turing machine in polynomial time.

Question 2 (15 marks)

Consider the following problem: Given a wine wholesaler has a wine barrel of capacity n litres and an array of prices that includes prices of wine volumes that are sold to restaurants of size smaller than n. Determine the maximum value obtainable by dividing up the wine and selling the smaller volumes.

Example 1: If the wine barrel capacity is 8 litres and the values of different volumes that can be sold are given as the follows, then the maximum obtainable value is 22 (by two wine volumes of 2 and 6)

Volume (litres)	1	2	3	4	5	6	7	8
Price	1	5	8	9	10	17	17	20

Example 2: If the prices are as follows, then the maximum obtainable value is 24 (by eight wine volume sales of 1 litre)

Volume (litres)	1	2	3	4	5	6	7	8
Price	3	5	8	9	10	17	17	20

a) Explain how many ways are there to get different volumes for sale from a wine barrel of capacity n litres? (2 marks)

k^n where k is the size of the price array, because there are k possible decisions for up to n places where we can choose to make a sales decision for k items.

b) Outline how a naïve Brute Force solution could to solve this problem. (2 marks)

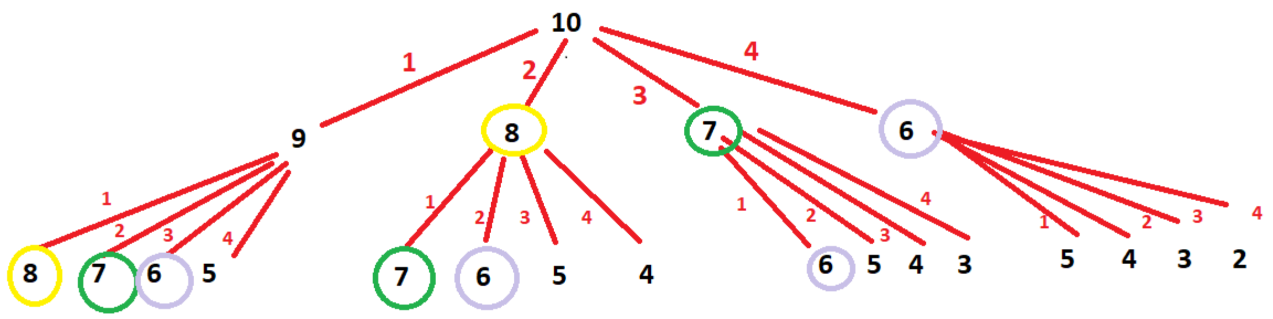
A naive solution to this problem is to generate all configurations of different wine volumes and find the highest-priced configuration. We can get the best price by taking different wine volumes out and comparing the values obtained after a sale. We can recursively call the same function for a wine volume obtained after a sale.

Let WS(n) be the required (best possible price) value for a wine sales from a barrel of volume n metres^3.

WS(n) can be written as follows.

WS(n) = max(price[i] + WS(n-i-1)) for all i in {0, 1 .. n-1}

c) If the wine barrel contains a volume of 10 litres and the volumes for sale are=[1,2,3,4] and the price=[1,5,8,9] show all the possible decisions using the naïve Brute Force method as a decision tree for 2 levels of decisions. (2 marks)



d) What features of this problem can be identified to solve it more efficiently? Explain these features and which design pattern can be used to improve time complexity. (2 marks)

We can see that there are many sub-problems that are solved again and again. This problem has the Overlapping dependent sub-problems property. Re-computations of the same sub-problems can be avoided by constructing a temporary array in a bottom-up manner and can efficiently be solved using Dynamic Programming design pattern.

e) Complete the more efficient solution for the wine sales problem using structured pseudocode below. (4 marks)

```
Algorithm wineSales(volumes, price, n)
    // volumes array. volumes for sale, volumes[1] is first element
    // price array, price[1] is the first element
    // n is the size of the volumes and price arrays

    // initialise the value array, where value[1] is the first element
    For i=1 to n do
      If (volumes[1] == 1) then
        // then all discrete volumes can be sold from 1 to n
        value[i] := i*price[1]
      Else
        value[i] := 0
      End if
    End do

    // Build the array of n elements value[]
    for i = 2 to n do
        for j = 1 to i do
            if (i – volumes[j]==0) then
                value[i] = maximum(value[i], 0 + price[j])
            else if (i – volumes[j] > 0) then
               value[i] = maximum(value[i], value[i-volumes[j]] + price[j])
            end if
        end do
    end do

end algorithm
```

f) State the time complexity and the space complexity of the solution from part e). Justify your responses. (2 marks)

A more efficient solution of time complexity O(n^2) due to the nested loop of approx. n iterations each and the space complexity O(n) for the wine sales problem using bottom up Dynamic Programming in an array.

g) With reference to the pseudocode in part e), in which ADT is the solution for wine sales from a wine barrel of n litres found? (1 mark)

Solution is in value[n] of the array after processing has been done.

Question 3 (12 marks)

Genevieve and Bacchus consider the following number puzzle problem:

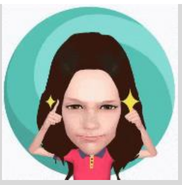
There is a three digit integer, let the number be abc. What is the largest three digit number with the property that the number is equal to the sum of its hundreds digit, the square of its ten digit and the cube of its units digit?	Written algebraically a solution for: $abc = a + b^2 + c^3$ where a, b, c are integers between 0 and 9

Genevieve does a little bit of algebra before trying to solve the problem with the following results.

$$\begin{aligned} 100a + 10b + c &= a + b^2 + c^3 \\ 99a + 10b - b^2 &= c(c^2 - 1) \\ 99a + b(10 - b) &= (c-1)c(c+1) \end{aligned}$$

And starts making a table to try out all the combinations

$99a$	$b(10-b)$	$(c-1)c(c+1)$
$99 \times 1 = 99$	$1 \times 9 = 9$	$1 \times 2 \times 3 = 6$
$99 \times 2 = 198$	$2 \times 8 = 16$	$2 \times 3 \times 4 = 24$
$99 \times 3 = 297$	$3 \times 7 = 21$	$3 \times 4 \times 5 = 60$
$99 \times 4 = 396$	$4 \times 6 = 24$	$4 \times 5 \times 6 = 120$
$99 \times 5 = 495$	$5 \times 5 = 25$	$5 \times 6 \times 7 = 210$
$99 \times 6 = 594$	$6 \times 4 = 24$	$6 \times 7 \times 8 = 336$
$99 \times 7 = 693$	$7 \times 3 = 21$	$7 \times 8 \times 9 = 504$
$99 \times 8 = 792$	$8 \times 2 = 16$	$8 \times 9 \times 10 = 720$
$99 \times 9 = 891$	$9 \times 1 = 9$	



a) Write a Generate-and-Test algorithm to solve this problem for Genevieve using the table algebra method above. (6 marks)

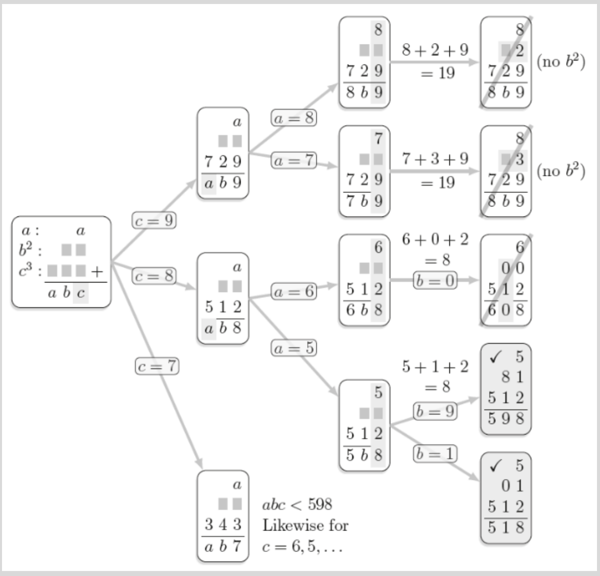
```
Algorithm Genevieve
Create List SolutionsList
// Generate all combinations
For a=1 to 9 do
       Term1 := 99*a
    For b=1 to 9 do
        Term2 := b*(10-b)
        For c=2 to 9 do
            Term3 := (c-1)*c*(c+1)
            // Add array of 6 values a,b,c,Term1,Term2,Term3 to SolutionsList
            Add [a,b,c,Term1,Term2,Term3] to SolutionsList
        End do
    End do
End do
// Test the combinations
For each item in SolutionsList do
      // a := item[1], b := item[2], c := item[3]
      // Term1 := item[4], Term2 := item[5], Term3 := item[6]
      If (item[4] + item[5] == item[6]) then
            Print Solution found for item[1], item[2], item[3]
      End if
End do
End Algorithm
```

1 Mark Structured pseudocode
1 Mark clarity
1 Mark Generate all combinations first
1 Mark test all combinations to find the answer
1 Mark correct use of ADTs
1 Mark correct logic

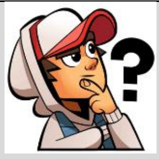
Question 3 (continued)

Bacchus has a different approach and tries to solve this problem by considering the possible values of b^2 and c^3. Where the number $abc = a + b^2 + c^3$

Digit	0	1	2	3	4	5	6	7	8	9
Square	0	1	4	9	16	25	36	49	64	81
Cube	0	1	8	27	64	125	216	343	512	729



Bacchus looks for conditional logic in solving this problem

	$a \neq 0, b \neq 0, c \neq 0$, since $abc \neq 0$ c can only be {8,9} a can take values from the first digit of c^3 to $(c - 1)$ The last digit of b^2 could be {1,4,5,6,9}

He starts by trying the largest possible values of c (c^3) first, then filling in possible values for a and then b (b^2).
This trial-and-error search is presented here as a tree with the dead-ends paths crossed out.

b) Write a Backtracking algorithm to solve this problem using Bacchus' methods. (6 marks)

Note: Conditional logic can be indicated using comments and does not need to be given in detail

```
Algorithm Bacchus
A := 0; B := 0
Create Stack ABCStack
// Start with C
For C=8 to 9 do
    Push [A, B, C, B, $C^3$] onto ABCStack ..// Candidate
End do
solutionFound := False

While (NOT solutionFound) AND (ABCStack is not empty) do
    Get [A, B, C, $B^2$,$C^3$] from top of ABCStack
    Pop ABCStack
    If (A == 0) then  // A has not been set, what A is possible - use Conditional Logic
        A := floor($C^3$/100)
        For i=A to (C-1) do
           Push [A, B, C, $B^2$,$C^3$] onto ABCStack  // Candidate Solution pushed onto stack
        End do
    Else If (A between 1 and 9) then // What B is possible - use Conditional Logic
        B :={ (Last digit of $C^3$+10)-C-A, B:= C-A} // Possible values of B using C-A arithmetic
        If (B in {1,4,5,6,9}) // could it be a perfect square $B^2$
            If (A*B*C = A + $B^2$ + $C^3$) then
                solutionFound := True
            End if
        End if
    End if
End do
End Algorithm
```

1 Mark Structured pseudocode
1 Mark clarity
1 Uses a stack to explore candidates and backtrack
1 Uses conditional logic to expand paths
1 Mark correct use of ADTs
1 Mark correct logic

Question 4 (10 marks)

a) Briefly explain what is DNA Computing. How does it differ from silicon computer technologies? (2 marks)

DNA computing is a form of computing which uses DNA, biochemistry, and molecular biology, instead of the traditional silicon-based computer technologies.
DNA computing, or more generally molecular computing, is based on manipulations with DNA strands to represent symbols and information using some basic biological transformations.

b) What are the benefits of DNA Computing? Explain the implications for solving NP problems. (2 marks)

DNA computing offers huge parallelism for computing, information data can be proceeded in parallel.
After that, a major goal of subsequent research is how to use DNA manipulations to solve NP, NP-Complete and NP-hard problems. DNA computing is a promising method for unconventional computation, owing to its merits of massive parallelism and efficiency in NP problem solving.

c) In 1998 Adelman conducted an experiment to solve the Hamiltonian path problem using DNA computing. Outline the main steps of Adelman's algorithm that was executed with DNA strands. (3 marks)

1. Encode the vertexes, edges and weights using DNA sequences representing the symbols.
2. From step1, generate random paths using vertex
3. From all paths created in step 2, keep only those that start at source s and end at target t.
4. From all remaining paths, keep only those that visit n vertices.
5. From all remaining paths, keep only those that visit each vertex once only.
6. If the path remaining more than 1 then From all remaining paths, keep only the shortlist path.
7. If any path remains return, “yes” with the path ;otherwise, return “no”.

To encode the solution to the Hamiltonian path problem Adelman worked with individual strands of DNA, the solutions formed quickly in parallel, thereafter several technical processes were needed to un-decipher the DNA solutions, anecdotally these processes took days.

d) What would be the space complexity and time complexity required for using DNA computing successfully for solving NP problems? Hence, what are the implications for encoding information and understanding solutions? (3 marks)

DNA computing needs to have an efficient degree of spatial complexity and time complexity in encoding the molecules by certain technical methods that will be used to build a molecular computer to hold the solution space for large problems.
One of the major problem for Adleman's DNA computing experiments, is the time involved in extracting and recombining DNA. While DNA processes within the test-tube can take place millions of times per second, extraction processes, whereby individual strands of DNA are manually isolated and spliced, can take several hours and even days, just for the simplest problems.
To apply DNA computing algorithms to the processes of NP- complete problem solving, a polynomial order of space complexity is required to store the information for problems and solutions found using DNA computers must be able to be extracted and understood in polynomial time.

Question 5 (5 marks)

A sequence of numbers is called a zig-zag sequence if the differences between successive numbers strictly alternate between positive and negative.

For example, 1,7,4,9,2,5 is a zig-zag sequence because the differences (6,-3,5,-7,3) are alternately positive and negative.	In contrast, 1,4,7,2,5 and 1,7,4,5,5 are not zig-zag sequences, the first because its first two differences are positive and the second because its last difference is zero.

Given a sequence of integers, sequence, write an algorithm in commented and structured pseudocode that returns the length of the longest subsequence of sequence in an array of length n that is a zig-zag sequence with O(n) time complexity.

```
function zigzagMaxLength(Array A)
    lenN := length(A) // A[1] is the first element of the array
    i := 2
    maxzigzag := 0  // keeps track of the maximum zig-zag sequence so far
    up := (A[i-1] > A[i]) // Boolean to show up or NOT up sequence for consecutive cells
    count := 1
    // one loop through array A[n] for O(n) time complexity
    while (i < lenN) do
        if (up and A[i] < A[i-1]) OR ((NOT up) AND (A[i] > A[i-1])) then
            count := count + 1
            up = NOT (up)
            // toggle the Boolean to look for the opposite way next time in this loop
        else
            // stopped zig-zagging sequence
            maxzigzag := maximum(maxzigzag, count)
            count := 0  //reset the count
        end if
        i := i + 1  // Move along the index for array A
     end do
     return maxzigzag
end function
```

1 Mark Structured pseudocode
1 Has one main loop to achieve O(n)
1 toggles up down logic for values
1 Mark correct use of ADTs
1 Mark correct logic

Question 6 (12 marks)

Consider the Bellman-Ford algorithm which works to find shortest paths in a weighted graph G.

```
Algorithm Bellman-Ford(G,v)
// Input G a weighted graph
// Input node v the source node, all nodes have attribute of distance

For each node n in G do
    n.distance := ∞   // initialise all nodes to have infinite distance
End do
v.distance := 0  // initialise source node to zero

For i=1 to |V|-1 do
    For each (u,w) in E do
        If u.distance + length(u,w) < w.distance then
             w.distance := u.distance + length(u.w)
        End if
    End do
End do

For each (u,w) in E do
    If u.distance + length(u,w) < w.distance then
        Print Negative cycle detected – shortest path not found
    End if
End do
End Algorithm
```

a) Fill in the missing parts of the algorithm. What is the missing action in the algorithm shown? (3 marks)

`|V|-1`

`w.distance := u.distance + length(u.w`

`u.distance + length(u,w) < w.distance`

b) What is the time complexity of the Bellman-Ford algorithm in terms of |V| and |E|? (1 mark)

O(|V||E|) due to the actions of the main nested loop.

c) If there are no negative cycles, what is the largest possible diameter of a graph G? (1 mark)

If there is no negative cycle, the length of any shortest path in terms of its edges is at most |V|-1. Therefore the largest possible diameter of a graph G is |V|-1

d) Describe any modifications to the ADTs and the pseudocode that would be needed to keep track of the predecessor node for the shortest path in the algorithm above? (3 marks)

An extra attribute previous is added to all nodes of graph G.
In the first loop where n.distance is initialised, also set node attribute n.previous to unknown
In the second nested loop when the distance is relaxed for edge u-w set node attribute w.previous to node u

Question 6 (continued)

e) Assuming that a graph G={V,E} has no negative cycles, using induction give a proof of correctness for the Bellman-Ford algorithm that includes a justification for the main loop running |V|-1 times.

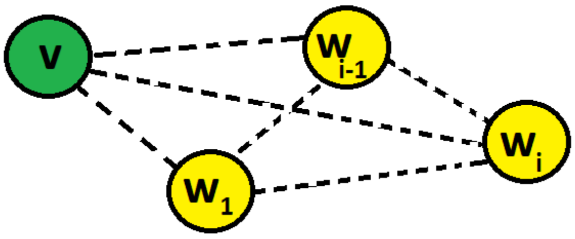
(4 marks)

Base case: If there are |V|=2 and |E|=1 connecting the two nodes, then Bellman-Ford will find the shortest path from the source node to the other node, since v.distance=0 is initialised for the source node and w.distance=∞ for all other nodes, the for i=1 to 1 runs for each edge u-v of which there is only one, the non-source node w will be updated w.distance := u.distance + length(u,w).

Assume that the path of P from v to w_{i-1} is a shortest path from v to w with at most $i-1$ edges after $i-1$ iterations of the main loop `For i=1 to |V|-1 do`

By induction on the next iteration of loop for any i then the node $\boldsymbol{u = w_{i-1}}$ where w_{i-1} is any node prior to w where $\boldsymbol{w = w_i}$ on shortest path P, then the distance after i iterations the $w_i.\text{distance} =$ w_i.distance+length(w_i, w_{i-1}) will be updated with the shortest possible P by checking of all possible paths from $\boldsymbol{u = w_{i-1}}$ to $\boldsymbol{w = w_i}$ `u.distance + length(u,w) < w.distance`

Since the main loop runs until i=|V|-1 , the value |V|-1 is the largest number of nodes that can possibly be in a path P in a the graph G of |V| nodes, and all possible shortest paths P to nodes $w_1, w_2, w_3, \ldots, w_{i-1}, w_i$ will be updated correctly starting from source node v



Question 7 (12 marks)

a) What is the definition of a computable function as defined in the context of mathematics and computer science? (2 mark)

The Church-Turing thesis defines a computable problem as one that is able to be solved by a Turing Machine given unbound resources of time and space.

b) The Entscheidungsproblem (German for "decision problem") is a challenge posed by Hilbert's Program. What is the definition and main aim of the posing this problem? (2 marks)

The problem asks for an algorithm as an effective method for determining decidability, so that it can consider, as input, a statement and answers "Yes" or "No" according to whether the statement is universally valid in satisfying axioms of the problem.

c) Briefly describe the Halting problem. Is the Halting problem decidable? Explain. (2 marks)

In computability theory, the halting problem is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running, or continue to run forever. Alan Turing proved in 1936 that a general algorithm to solve the halting problem for all possible program-input pairs cannot exist. The Halting problem is undecidable over Turing machines and is therefore unsolvable.

d) Outline the contradiction using logic/pseudocode or a diagram that can be set up to show the decidability of the Halting problem. (3 marks)

A halting function h is defined
The halting function at the code of another program and decide if that other program will ever stop running?

$$h(p,i) = \begin{Bmatrix} 1 & if\ h(p,i) = halts \\ 0 & if\ h(p,i) = loops \end{Bmatrix}$$

A halting function h which reads as input a program P and an input I, and determines correctly if P will halt on input I, output Yes, P will stop given I or No, P will never stop given I

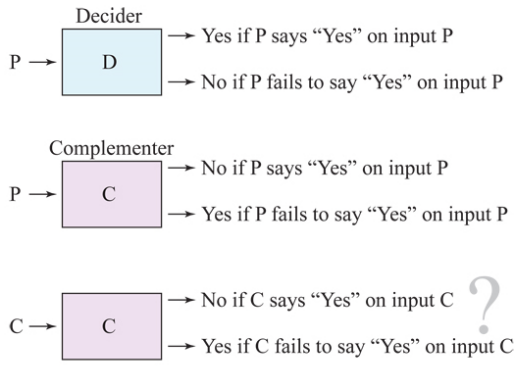
If we build a function g that sets up a contradiction

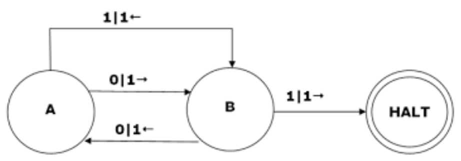
$$g(i) = \begin{Bmatrix} loops & if\ h(i,i) = halts \\ halts & if\ h(i,i) = loops \end{Bmatrix}$$

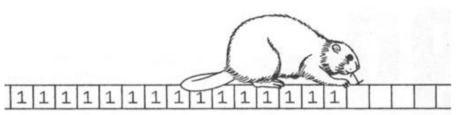
From the definition of g only one of the following holds. What happens when g runs with input g?

$$g(g(i)) = \begin{Bmatrix} loops & if\ h(g(i),g(i)) = halts \\ halts & if\ h(g(i),g(i)) = loops \end{Bmatrix}$$

Whatever h will say will happen, g says the opposite. So h could not have correctly predicted what will happen


Decider
P
D
Yes if P says "Yes" on input P
No if P fails to say "Yes" on input P
Complementer
P
C
No if P says "Yes" on input P
Yes if P fails to say "Yes" on input P
C
C
No if C says "Yes" on input C
Yes if C fails to say "Yes" on input C
?





e) Briefly describe the Busy Beaver problem. Is the Busy Beaver problem decidable? Explain.(3 marks)

Given an n-state Turing Machine with a two symbol alphabet {0, 1}, what is the maximum number of 1's that the machine can print on an initially blank tape (filled with 0's) before halting?

Currently it has been solved for small numbers of states such as n=1, 2, 3, 4 by reasoning about and running all possible Turing Machines, but it can't be solved in general as for large number of states the outputs are uncountable and unknown. Theorists call such problems non-computable and therefor undecidable.

END OF TRIAL EXAM