M269 Exam 2019

Prsntn 2018J Exam

Contents

1 Prsntn 2018J Exam Qs

1.1 M269 2018J Exam Qs

- M269 Algorithms, Data Structures and Computability
- Presentation 2018J Exam
- Date Friday, 14 June 2019 Time 14:30–17:30
- There are **TWO** parts to this examination. You should attempt all questions in **both** parts
- **Part 1** carries **65 marks** — **80 minutes**
- **Part 2** carries **35 marks** — **90 minutes**
- **Note** see the original exam paper for exact wording and formatting — these slides and notes may change some wording and formatting
- **Note** The 2015J exam and before had Part 1 with 60 marks (100 minutes), Part 2 with 40 marks (70 minutes)

Go to Solns

1.2 M269 2018J Exam Q Part1

- Answer every question in this part.
- The marks for each question are given below the question number.
- Answers to questions in this Part should be written **on this paper** in the spaces provided, or in the case of multiple-choice questions you should tick the appropriate box(es).
- If you tick **more** boxes than indicated for a multiple choice question, you will receive **no** marks for your answer to that question.
- Use the provided answer books for any rough working.

Go to Soln Part1

1.3 M269 2018J Exam Q 1

- Which **one** of the following statements is **true**? (Tick **one** box.)

A. A specification of an algorithm in structured English is a formal description of the inputs and outputs for that algorithm.

B. A problem is computable if it can be expressed as an algorithm.

C. An algorithm will always terminate.

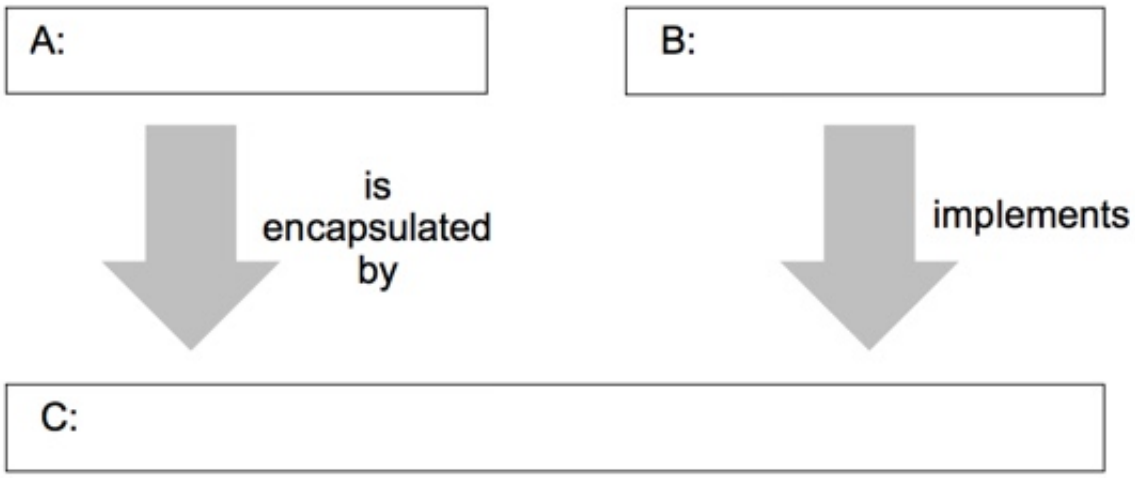
D. A problem for which the result is yes or no, is a decision problem.

Go to Soln 1

1.4 M269 2018J Exam Q 2

- Complete the following diagram by filling the empty boxes labelled **A**, **B** and **C** with the appropriate terms from the following list:

Abstract Data Type, *Data*, *Data Structure*



Go to Soln 2

1.5 M269 2018J Exam Q 3

- You are given two data structures. The first is an unordered list of integer values without duplicates, the second is an ordered list of integer values without duplicates.
- Here are examples of such lists, you will not need to use these lists to answer the question:

Unordered List: [17, 99, 25, 31, 39, 77, 83, 89, 41, 52, 67, 69, 2, 16, 91]
Ordered List: [2, 16, 17, 25, 31, 39, 41, 52, 67, 69, 77, 83, 89, 91, 99]

- For a list with n items

(a) What are the best-case and worst-case Big-O complexities of a linear search over an un-ordered list?

(b) What are the best-case and worst-case Big-O complexities of a binary search over an ordered list?

	Best case Big-O	**Worst case Big-O**
Linear search over unordered list with n items		
Binary search over ordered list with n items		

- In each of the four cases, briefly explain your answer:

Go to Soln 3

1.6 M269 2018J Exam Q 4

- A Python program contains a loop with the following guard

```
while (not (i == 3) or not (j < 2)) and (i == 3):
```

- Complete the following truth table, where:

  P represents `i == 3`

  Q represents `j < 2`

P	Q	¬P	¬Q	¬P ∨ ¬Q	(¬P ∨ ¬Q) ∧ P
F	F				
F	T				
T	F				
T	T				

- Use the results from your truth table to choose which one of the following statements is true about the above guard. (Tick **one** box.)

A. It simplifies to `not (j < 2)`.

B. The loop will never be entered.

C. The loop will never terminate.

D. It simplifies to `(i == 3)` and `not (j < 2)`.

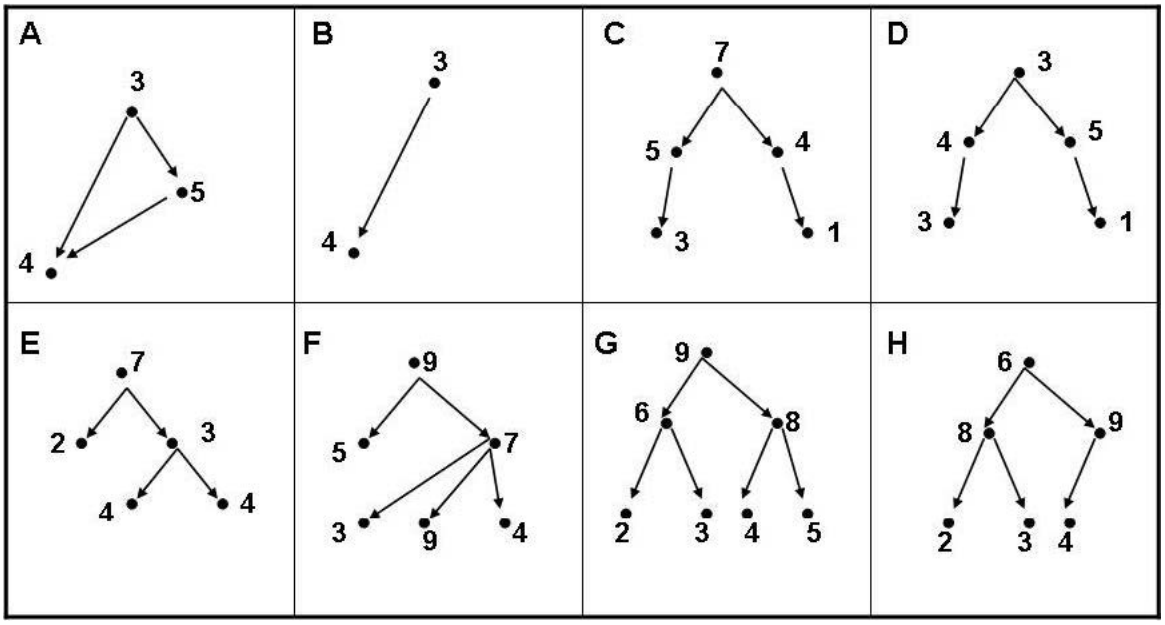
E. It simplifies to `i == 3`

Go to Soln 4

1.7 M269 2018J Exam Q 5

- Consider the diagrams in A–H, where nodes are represented by dots and edges by arrows. The numbers are the keys for the corresponding nodes.

  Note that in diagrams C and D the arrows in the bottom level show leaf 3 to be to the left of its parent node (a left branch), and leaf 1 to be to the right of its parent node (a right branch).



- On each line, write one or more letters, or write "None".

(a) Which of **A**, **B**, **E** and **F**, if any, are **not** a tree?

(b) Which of **E**, **F**, **G** and **H**, if any, are binary trees?

(c) Which of **B**, **C**, **F** and **G**, if any, are complete binary trees?

(d) Which of **A**, **D**, **E** and **H**, if any, are a heap?

Go to Soln 5

1.8 M269 2018J Exam Q 6

- Consider the following function, which takes a non-negative integer as an argument.

```
def evaluate(n):
  a = 1
  b = 0
  c = 0
  for i in range (n):
    a = a + i
    for j in range (n):
      a = a + j
      b = b + a
      for k in range (n):
        a = a + k
        b = b + a
        c = c + a + b
  resolution = (c * b) / a
  return resolution
```

- From the options below, select the two that represent the correct combination of T(n) and Big-O complexity for this function.

  You may assume that a step (i.e. the basic unit of computation) is the assignment statement. **(6 marks)**

  (Tick **one** box for T(n) and **one** box for Big-O complexity.)

A. $T(n) = 3n^3 + 2n^2 + n + 4$	I. $O(n)$
B. $T(n) = 3n^3 + 2n^2 + n + 3$	II. $O(n^2)$
C. $T(n) = 3n^2 + 2n + 3$	III. $O(3n^2)$
D. $T(n) = 3n^2 + 2n + 3$	IV. $O(n^3)$
E. $T(n) = n + 10$	V. $O(3n^3)$

- Explain how you arrived at T(n) and the associated Big-O complexity.

Go to Soln 6

1.9 M269 2018J Exam Q 7

(a) Which **one** of the following statements about hash tables is **true**? (Tick **one** box.) **(4 marks)**

A. Hash tables don't always require key values to be unique.

B. Linear probing always examines every storage location in a hash table.

C. The load factor is a measure of how many collisions have occurred when using a hash table.

D. A collision occurs when two key values map to the same location in a hash table.

(b) Calculate the **load factor** for the hash table below. Show your working. **(4 marks)**

	Germaine			Anu	Eric			
0	1	2	3	4	5	6	7	8

Go to Soln 7

1.10 M269 2018J Exam Q 8

a) Draw the binary search tree that results from the insertion of the following values, in order, in an initially empty tree:

55, 59, 34, 29, 86, 65, 68.

b) For the tree you produce, annotate each node with its balance factor and so explain whether the tree you produced is balanced or unbalanced. **(6 marks)**

Go to Soln 8

1.11 M269 2018J Exam Q 9

(a) In Python, a dictionary of dictionaries can be used to represent a directed graph's adjacency lists.

```
Directedgraph1 = {
     0: {'neighbours': [1,4] },
     1: {'neighbours': [3,4] },
     2: {'neighbours': []    },
     3: {'neighbours': [0]   },
     4: {'neighbours': [2,1] },
     5: {'neighbours': []    }
}
```

Draw the graph that corresponds to the adjacency list given above.

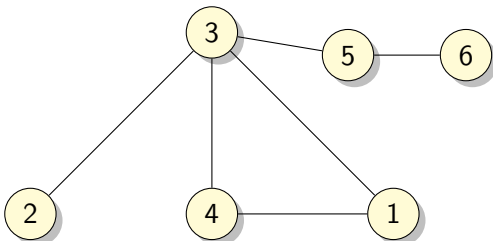
(b) Complete the following statements about your graph.

- The graph is _____ (choose from *Cyclic/Acyclic*) because: *(insert your answer here)*
- The graph is _____ (choose from *Sparse/Dense*) because: *(insert your answer here)*

Go to Soln 9

1.12 M269 2018J Exam Q 10

- Consider the following undirected graph:



- *[In the space below,]* draw one of the spanning trees that could be generated from a **Breadth** First Search (BFS) of the above graph, starting at vertex 1:

Go to Soln 10

1.13 M269 2018J Exam Q 11

- In propositional logic, what does it mean to say that two well-formed formulae are *equivalent*?
- Are the following well-formed formulae equivalent to each other?

$$(P \wedge \neg Q) \quad \neg(\neg P \vee Q)$$

- Justify your answer by showing appropriate truth tables.

Go to Soln 11

1.14 M269 2018J Exam Q 12

- Consider the following particular interpretation $\mathcal{I}$ for predicate logic allowing facts to be expressed about about boats, their homeport and the ports at which they have docked.
- The domain of individuals is $\mathcal{D}$ = {Dover, Poole, Portsmouth, Plymouth, Endeavour, Serenity, Adagio}.
- The constants *dover*, *poole*, *portsmouth*, *plymouth*, *endeavour*, serenity and adagio are assigned to the corresponding elements.
- Two predicate symbols are assigned binary relations as follows:
- $\mathcal{I}$(*homeport*) = {(Endeavour,Dover), (Serenity,Plymouth), (Adagio, Dover)}
- $\mathcal{I}$(*has_docked_at*) = {(Endeavour,Poole), (Endeavour,Dover), (Endeavour, Plymouth), (Serenity, Plymouth), (Serenity, Portsmouth), (Adagio, Dover), (Adagio, Plymouth)}
- For both relations, the first element of each tuple is the boat's name and the second is the port's name

(a) Consider the English sentence:

There is at least one boat that has a home port of Dover and that has docked at Poole.

Write this as a well-formed predicate logic formula.

(b) This formula is _____ *(choose from TRUE/FALSE)*, under the interpretation given above.

(c) Explain your answer to (b) *[in the box below]*.

You need to consider any relevant values for the variables, and show, using the domain and interpretation on the previous page, whether they make the formula TRUE or FALSE.

In your explanation, make sure that you use formal notation.

For example, instead of stating *Serenity hasn't docked at Poole* you would need to write (Serenity, Poole) $\notin \mathcal{I}(has\_docked\_at)$

(d) Give an appropriate English translation of the well-formed formula:

$\forall X.(\neg \text{has\_docked\_at}(X, poole) \rightarrow homeport(X, dover))$

Go to Soln 12

1.15 M269 2018J Exam Q 13

- The interpretation of the previous question can also be represented by a database containing the following tables and data:

homeport

boatname	home
Serenity	Plymouth
Adagio	Dover
Endeavour	Dover

has_docked_at

name	port
Endeavour	Poole
Endeavour	Dover
Endeavour	Plymouth
Serenity	Plymouth
Serenity	Portsmouth
Adagio	Dover
Adagio	Plymouth

(a) For the following SQL query, give the table returned by the query.

```
SELECT home, boatname
FROM homeport CROSS JOIN has_docked_at
WHERE home = port AND boatname = name;
```

(b) Write the request that the above query is answering.

Go to Soln 13

1.16 M269 2018J Exam Q 14

- Consider the statements (numbered 1–4) below and write the letter of the most appropriate term (labelled A–E) into the underlined empty spaces.

1. A decision problem that is not computable is: _____
2. A tractable decision problem is: _____
3. A computational problem with co-domain {0,1} is: _____
4. The inability to re-write a program and know if it behaves the same as the original program is a consequence of: _____

- List of terms:

A. a decision problem,

B. the totality problem,

C. an undecidable problem,

D. the equivalence problem,

E. a class P problem.

Go to Soln 14

1.17 M269 2018J Exam Q 15

- Consider the following computational problems (numbered 1–5):

1. Deciding whether or not a given integer is even.
2. Finding the largest value in an unsorted list of integers.
3. Deciding whether a given algorithm will terminate for every possible input.
4. The decision version of the travelling salesman problem.
5. The equivalence problem.

- On each line below, write one or more of the above problem numbers, or write *None*.

A. Which problems, if any, **are computable**?

B. Which problems, if any, **are definitely in the class P**?

C. Which problems, if any, **are in the class NP**?

D. Which problems, if any, **are NP-complete**?

Go to Soln 15

1.18 M269 2018J Exam Q Part2

- Answer every question in this Part.
- Answers to questions in this Part must be written in the separate **answer books**, which you should also use for your rough working.

Go to Soln Part2

1.19 M269 2018J Exam Q 16

- The Stack Abstract Data Type describes a data structure in which items are available in last-in, first-out order.
- Items are pushed onto the top of a stack and popped off the top.
- The ADT for a Stack includes these operations:

- `new`, which creates a new, empty stack.
- `push`, which adds an item to the top of a stack.
- `pop`, which removes and returns the top item from the stack.
- `is_empty`, which tests to see if the stack is empty

(a) The following stand-alone Python function accepts a stack and returns a list with the stack's items but in reverse order. **(7 marks)**

- It assumes the ADT is implemented as a Python class, i.e. using `Stack()` as the `new` operation.

```
def reverse(in_stack):
  inverted_stack = Stack()
  while not in_stack.is_empty():
    inverted_stack.push(in_stack.pop())
  reversed_list = []
  while not inverted_stack.is_empty():
    reversed_list.append(inverted_stack.pop())
  return reversed_list
```

(i) What is the Big-O complexity of the `reverse` function for an input stack with n items? Explain your answer. Take each stack and list operation as a basic step of computation.

(ii) If the `reverse` operation is added to the Stack ADT it would be a modifier. Explain why.

(iii) What would you change in order to make the `reverse` operation an inspector? Don't write code, just describe the changes needed.

(b) A list is a palindrome if the list is the same as its reverse. **(8 marks)**

- So, `[1,2,3,2,1]` is a palindrome, `['a', 'b', 'b', 'a']` is a palindrome, `[1]` is a palindrome, `[['Fred'],['Freda'], ['Freya']]` is not a palindrome.

(i) Specify the problem of deciding if a given list is a palindrome. You may write the specification in English and/or formally with mathematical notation.

Name: palindrome

Inputs:

Preconditions:

Outputs:

Postconditions:

- Give your initial insight for an algorithm that uses a stack to solve the problem.
- You must use some or all of the Stack ADT's operations (`pop`, `push`, `new`, and `is_empty`).
- You can also use any of the basic list operations:
- create an empty list, append an item, calculate the length, access an item by position (indexing), iterate over the items or over the indices, check if two lists are equal.

(c) A training company, TComp, offers short courses to members of the public. **(5 marks)**

- Someone seeking information on a course on the TComp website will be shown a list of *prior knowledge courses* (which we will abbreviate to *prior courses* from now on).
- TComp needs to hold information about the courses, and which are prior courses for each other.
- So, course X has prior courses A, B and C, is TComp's way of saying that to do course X you should already know about the content of courses A and B and C.
- Those prior courses may also have prior courses, and a course may be a prior course for more than one other course.
- When a customer applies for a place on a course, they complete an application form on the website.
- This is sent electronically to a processing centre in which the applications are processed in the order they arrive.
- Courses have a limited number of places, so places are allocated to each course in the order the processed applications are received until the course is full.
- Any remaining applications are stored in case vacancies arise later.
- State, and briefly justify, the choice of abstract data types that you could adopt or adapt to represent:

(i) the course descriptions and the links to their prior knowledge courses;

(ii) the applications submitted for places on the courses.

- State explicitly any assumptions you make.

Go to Soln 16

1.20 M269 2018J Exam Q 17

- A TV programme is planning to screen a public lecture on aspects of Computability.
- To help shape the lecture, you've been asked to prepare a short report for the producers on the topics **decision problems** and **undecidability** with a particular focus on **the equivalence problem**.
- You should assume that the producers do not have a background in computer science and that the programme's intended audience is the general public.
- Your report **must** have the following structure:

1. A suitable title and a short paragraph defining computability.
2. A paragraph introducing decision problems.
3. A paragraph in which you describe the issue of undecidability.
4. A paragraph describing how undecidability relates to the equivalence problem.
5. A summary that describes why the equivalence problem is important in computing.

- Some marks will be awarded for a clear coherent text that is appropriate for its audience, so avoid unexplained technical jargon and abrupt changes of topic, and make sure your sentences fit together to tell an overall *story*.

- As a guide, you should aim to write roughly two to five sentences per paragraph.

Go to Soln 17

2 Prsntn 2018J Exam Solns

2.1 M269 2018J Exam Solns

- The solutions given below are not official solutions
- For some questions, alternatives are given — a student would only have to provide one

Go to Qs

2.2 M269 2018J Exam Soln Part1

- Part 1 solutions

Go to Q Part1

2.3 M269 2018J Exam Soln 1

A. A specification of an algorithm in structured English is a formal description of the inputs and outputs for that algorithm. **No** structured English is not formal

B. A problem is computable if it can be expressed as an algorithm. **No** the algorithm must terminate in a finite number of steps

C. An algorithm will always terminate. **No** it could loop

D. A problem for which the result is yes or no, is a decision problem. **Yes**

Go to Q 1

2.4 M269 2018J Exam Soln 2

- A. Data (is encapsulated by C.)
- B. Data Structure (implements C.)
- C. Abstract Data Type

Go to Q 2

2.5 M269 2018J Exam Soln 3

(a) Linear search best case O(1) first item

- Linear search worst case O(n) last item or not present

(b) Binary search best case O(1) middle item

- Binary search worst case O(log n) depth of search tree — the search space is halved on each iteration

Go to Q 3

2.6 M269 2018J Exam Soln 4

P	Q	¬P	¬Q	¬P ∨ ¬Q	(¬P ∨ ¬Q) ∧ P	P ∧ ¬Q
F	F	T	T	T	F	F
F	T	T	F	T	F	F
T	F	F	T	T	T	T
T	T	F	F	F	F	F

- Extra column for the second part of the question

D. It simplifies to `(i == 3)` and `not (j < 2)`.

Go to Q 4

2.7 M269 2018J Exam Soln 5

(a) Which of **A**, **B**, **E** and **F**, if any, are **not** a tree? **A**

(b) Which of **E**, **F**, **G** and **H**, if any, are binary trees? **E**, **G** and **H**

(c) Which of **B**, **C**, **F** and **G**, if any, are complete binary trees? **B**, **G**

(d) Which of **A**, **D**, **E** and **H**, if any, are a heap? **None**

Go to Q 5

2.8 M269 2018J Exam Soln 6

A. $T(n) = 3n^3 + 2n^2 + n + 4$

IV. $O(n^3)$

- There are 4 assignments outside all loops at lines 2, 3, 4 and 14
- There is 1 assignment at line 6 in the outer loop (the loop indexed by `i` at line 5) — n assignments
- There are 2 assignments at lines 8 and 9 in the first nested loop (the loop indexed by `j` at line 7) — $2n^2$ assignments
- There are 3 assignments at lines 11, 12 and 13 in the second nested loop (the loop indexed by `k` at line 10) — $3n^3$ assignments
- Code with assignments

```
def evaluate(n):                    # Assignments
  a = 1                             # 1
  b = 0                             # 1
  c = 0                             # 1
  for i in range (n):               # n for loop
    a = a + i                       #   1×n
    for j in range (n):             # 2n^2 for loop
      a = a + j                     #   1×n×n
      b = b + a                     #   1×n×n
      for k in range (n):           # 3n^3 for loop
        a = a + k                   #   1×n×n×n
        b = b + a                   #   1×n×n×n
        c = c + a + b               #   1×n×n×n
  resolution = (c * b) / a          # 1
  return resolution
```

Go to Q 6

2.9 M269 2018J Exam Soln 7

A. Hash tables don't always require key values to be unique. False

B. Linear probing always examines every storage location in a hash table. False

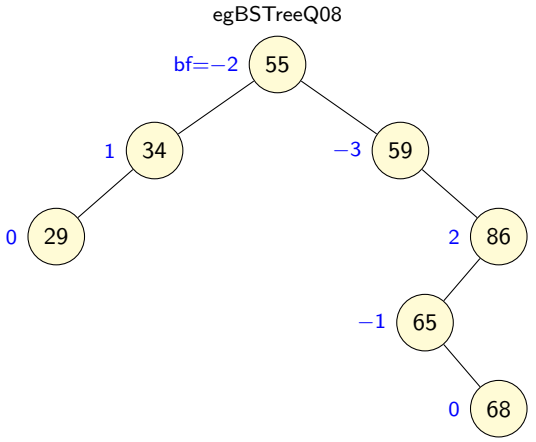
C. The load factor is a measure of how many collisions have occurred when using a hash table. False

D. A collision occurs when two key values map to the same location in a hash table. True

- The **load factor** is 3/9 (simplifies to 1/3)
- The **load factor** is defined as occupied slots/total slots — note the index of the slots starts at 0 so there are 9 slots (not 8)

Go to Q 7

2.10 M269 2018J Exam Soln 8

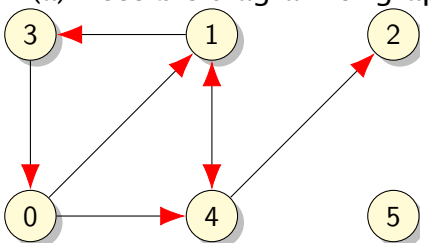


- The tree is not balanced since (several) nodes have balance factors outside {-1,0,1}

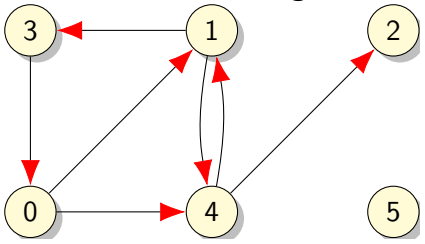
Go to Q 8

2.11 M269 2018J Exam Soln 9

(a) Possible diagram of graph:



- Here we have denoted two edges as one bi-directional — the following gives an alternative
- Alternative diagram with all edges as uni-directional single arrows



- Note that many equivalent diagrams are possible

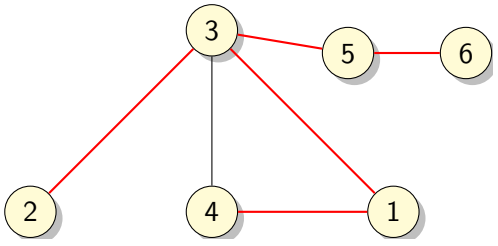
(b) The graph is *cyclic* since we have {1,4,1}, {0,1,3,0} and {0,4,1,3,0} as cycles

- This is a *sparse graph* — a dense graph is a graph in which the number of edges is close to the maximal number of edges — in a directed graph this is n(n – 1) — this would be 30 in this case which is a lot higher than the number of edges, 7 — hence this is a sparse graph

Go to Q 9

2.12 M269 2018J Exam Soln 10

- Spanning tree shown in thick, red
- Generated from a **Breadth** First Search (BFS) of the above graph, starting at vertex 1:



Go to Q 10

2.13 M269 2018J Exam Soln 11

- Two WFFs are *equivalent* if they have the same truth values for every interpretation.

P	Q	¬P	¬Q	(P ∧ ¬Q)	¬(¬P ∨ Q)
T	T	F	F	F	F
T	F	F	T	T	T
F	T	T	F	F	F
F	F	T	T	F	F

- You could also justify your answer using the logical equivalences for negation and the De Morgan rules (though the question explicitly asks for truth tables)

Go to Q 11

2.14 M269 2018J Exam Soln 12

(a) *There is at least one boat that has a home port of Dover and that has docked at Poole.*

- ∃X.(homeport(X, dover) ∧ has_docked_at(X, poole))

(b) **True**

(c) (Endeavour, Dover) ∈ $\mathcal{I}$(homeport)

∧ (Endeavour, Poole) ∈ $\mathcal{I}$(has_docked_at)

(d) ∀X.(¬has_docked_at(X, poole) → homeport(X, dover))

- If a boat has not docked at Poole has home port of Dover
- Any boat that has not docked at Poole then it has home port of Dover

Go to Q 12

2.15 M269 2018J Exam Soln 13

(a)

home	boatname
Dover	Endeavour
Plymouth	Serenity
Dover	Adagio

(b) State the homeport and boat name of every boat that has docked at its homeport?

Go to Q 13

2.16 M269 2018J Exam Soln 14

1. A decision problem that is not computable is: **C: an undecidable problem,**
2. A tractable decision problem is: **E: a class P problem.**
3. A computational problem with co-domain {0,1} is: **A: a decision problem,**

4. The inability to re-write a program and know if it behaves the same as the original program is a consequence of: **D: the equivalence problem,**

Go to Q 14

2.17 M269 2018J Exam Soln 15

1. Deciding whether or not a given integer is even. **computable**, **class P**, **class NP**
2. Finding the largest value in an unsorted list of integers. **computable**, **class P**, **class NP**
3. Deciding whether a given algorithm will terminate for every possible input. **not computable**
4. The decision version of the travelling salesman problem. **computable**, **NP-complete**, **class NP**
5. The equivalence problem. **not computable**

A. Which problems, if any, **are computable**? 1, 2, 4

B. Which problems, if any, **are definitely in the class P**? 1, 2

C. Which problems, if any, **are in the class NP**? 1, 2, 4

D. Which problems, if any, **are NP-complete**? 4

Go to Q 15

2.18 M269 2018J Exam Soln Part2

- Part 2 solutions

Go to Q Part2

2.19 M269 2018J Exam Soln 16

(a) (i) O(n) — two separate loops each with n operations

- There are two operations outside the loops, two operations inside each of the two loops and one operation in each of the `while` conditions so T(n) = 6n + 2 giving O(n)
- If you missed the operations in the `while` conditions then you would have T(n) = 4n + 2 still giving O(n)
- See What is the time complexity of Python List Reverse?, Complexity of Python Operations
- (ii) Modifier since `pop()` is destructive — on completion of the function, the stack is empty since all items are popped off it
- (iii) To make `reverse` a modifier, work on a copy of the input and return the result

(b) (i) **Name**: palindrome

Inputs: List of items $xs = \{x_1, x_2, \ldots, x_n\}$

Preconditions: The items can be compared for equality

Outputs:b, a Boolean

Postconditions: b is True if and only if the input is equal to its reverse, that is, a palindrome

(ii) Create empty stack s with new

Loop over the list and push each item onto the stack

Loop over the list and use pop to compare each item with the item the same distance from the end

Terminate the loop if the items are not equal (return False) or we get to the empty list (return True)

(c) (i) Represent courses as map from course code to pair of course description (string) and prior courses (as set of course codes) — the course code is a key

- This is equivalent to representing courses as a Directed Acyclic Graph (DAG) with nodes as courses (with descriptions) and edges indicating prior courses

(ii) Represent applications as list of application since the order matters

- Alternatively represent applications as a queue of application, since this would aid adding and processing order

Go to Q 16

2.20 M269 2018J Exam Soln 17

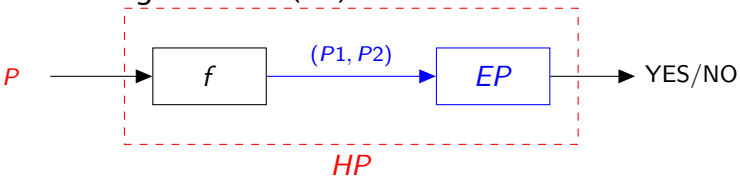
- **Title** Equivalence Problem: Does Not Compute
- Define Computability (see also Computability and Complexity) — a problem is computable if it can be solved in an effective manner by some algorithm.
- In the 1930s Gödel, Church and Turing provided formal models of computation — Turing machines and Lambda Calculus being the better known (but there are many others)
- Turing showed that although they look quite different, the models have exactly the same computational power
- A decision problem is a problem that can be posed as a yes-no question of the input values
- A decision problem is decidable if there exists an effective method for deriving the correct answer
- Some decision problems are undecidable:
- Halting problem — is there a program that can determine if any arbitrary program will halt or continue forever ?
- Proof by contradiction — Assume we have such a program (Turing Machine) h(f,x) that takes a program f and input x and determines if it halts or not

```
def halts(f,x):
  if f(x) terminates :
    return True
  else :
    return False
```

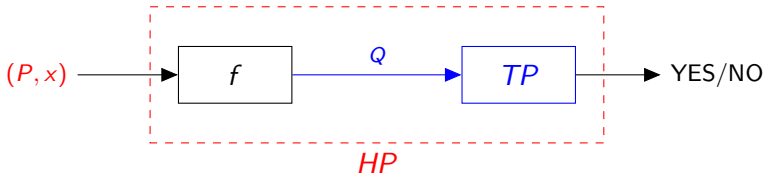
- We now use `halts`to define another function `r(f)` that that loops forever if `halts(f,f)` returns `True` otherwise it terminates

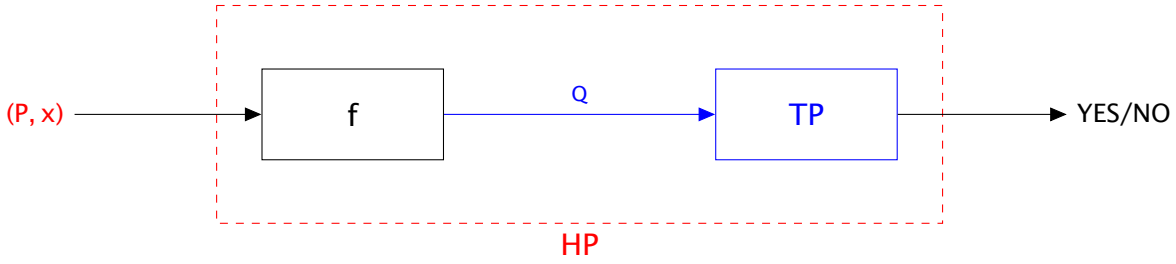
```
def r(f) :
  if halts(f,f) :
    while True :
      continue
  else :
    return
```

- We now ask what happens if we evaluate `r(r)` — if it terminates, it goes into a loop, and if it doesn't terminate then it stops
- Contradiction — the only thing that can be wrong is our assumption that `halts(f,x)` exists
- **Equivalence Problem**
  - Input: two programs P1 and P2
  - Output: YES if P1 and P2 solve the same problem (same output for same input) else NO
- We show that the Equivalence Problem (EP) is not computable by reduction from the Halting Problem (HP)



- The exam would probably not require a proof that the Equivalence Problem is undecidable but here is a proof via the Totality Problem
- **Totality Problem**
  - Input: program Q
  - Output: YES if Q terminates for all inputs else NO
- Assume we have algorithm TP to solve the Totality Problem
- Now reduce the Halting Problem to the Totality Problem

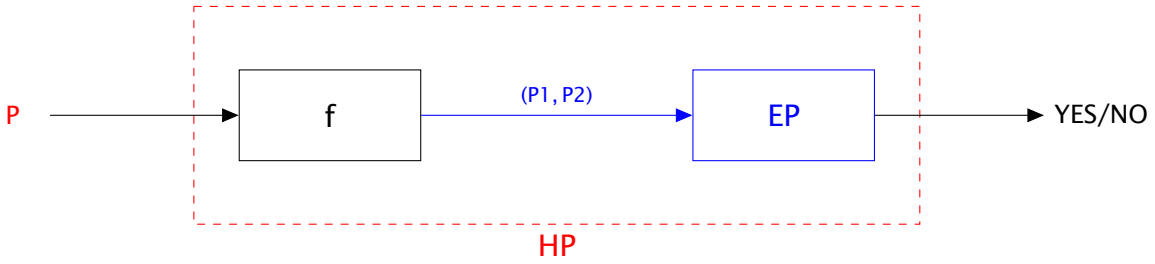




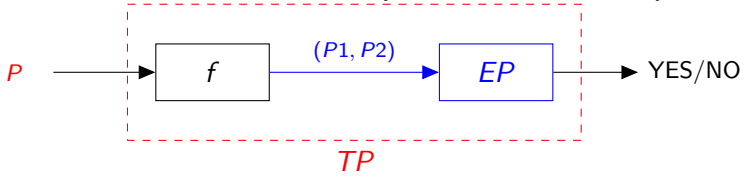
- Define f to transform inputs to HP to TP pseudo-Python

```
def f(P,x) :
  def Q(y):
    # ignore y
    P(x)
  return Q
```

- Run TP on Q
  - If TP returns YES then P halts on x
  - If TP returns NO then P does not halt on x
- We have *solved* the Halting Problem — contradiction



- **Equivalence Problem**
  - Input: two programs P1 and P2
  - Output: YES if P1 and P2 solve the same problem (same output for same input) else NO
- Assume we have algorithm EP to solve the Equivalence Problem
- Now reduce the Totality Problem to the Equivalence Problem



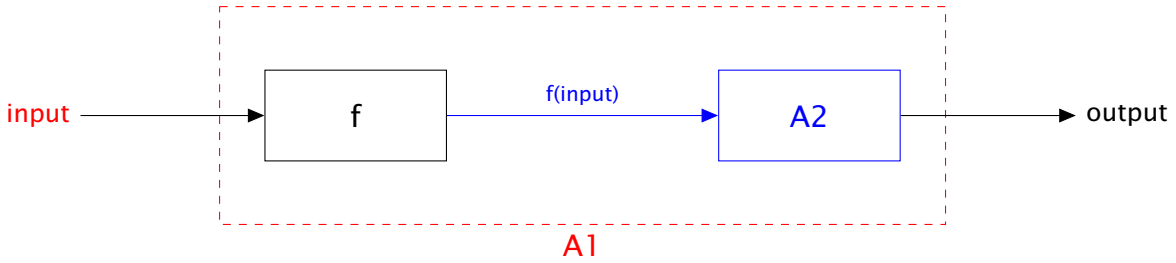
- Define f to transform inputs to TP to EP pseudo-Python

```
def f(P) :
  def P1(x):
    P(x)
    return "Same string"
  def P2(x)
    return "Same string"
```

```
    return (P1,P2)
```

- Run EP on (P1, P2)
  - If EP returns YES then P halts on all inputs
  - If EP returns NO then P does not halt on all inputs
- We have *solved* the Totality Problem — contradiction

Go to Q 17



- Rice's Theorem all non-trivial, semantic properties of programs are undecidable. H G Rice 1951 PhD Thesis
- Equivalently: For any non-trivial property of partial functions, no general and effective method can decide whether an algorithm computes a partial function with that property.
- A property of partial functions is called trivial if it holds for all partial computable functions or for none.
- Rice's Theorem and computability theory
- Let S be a set of languages that is nontrivial, meaning
  - there exists a Turing machine that recognizes a language in S
  - there exists a Turing machine that recognizes a language not in S
- Then, it is undecidable to determine whether the language recognized by an arbitrary Turing machine lies in S.
- This has implications for compilers and virus checkers
- Note that Rice's theorem does not say anything about those properties of machines or programs that are not also properties of functions and languages.
- For example, whether a machine runs for more than 100 steps on some input is a decidable property, even though it is non-trivial.
- Note: Rice's Theorem is not mentioned in the M269 material so the exam would only expect reference to implications for virus checkers and automated program verification

Go to Q 17