MONASH University

Semester Two 2019
Examination Period

Faculty of Information Technology

EXAM CODES:	FIT1008
TITLE OF PAPER:	Introduction to computer science
EXAM DURATION:	3 hours 10 mins

Rules

During an exam, you must not have in your possession any item/material that has not been authorised for your exam. This includes books, notes, paper, electronic device/s, mobile phone, smart watch/device, calculator, pencil case, or writing on any part of your body. Any authorised items are listed below. Items/materials on your desk, chair, in your clothing or otherwise on your person will be deemed to be in your possession.

You must not retain, copy, memorise or note down any exam content for personal use or to share with any other person by any means following your exam.

You must comply with any instructions given to you by an exam supervisor.

As a student, and under Monash University's Student Academic Integrity procedure, you must undertake your in-semester tasks, and end-of-semester tasks, including exams, with honesty and integrity. In exams, you must not allow anyone else to do work for you and you must not do any work for others. You must not contact, or attempt to contact, another person in an attempt to gain unfair advantage during your exam session. Assessors may take reasonable steps to check that your work displays the expected standards of academic integrity.

Failure to comply with the above instructions, or attempting to cheat or cheating in an exam may constitute a breach of instructions under regulation 23 of the Monash University (Academic Board) Regulations or may constitute an act of academic misconduct under Part 7 of the Monash University (Council) Regulations.

Authorised Materials

OPEN BOOK	☐ YES	☑ NO
CALCULATORS	☐ YES	☑ NO
SPECIFICALLY PERMITTED ITEMS	☐ YES	☑ NO

if yes, items permitted are:

Instructions

Please answer all questions.

This exam is worth 60% of your overall mark.

To answer a question that requires a code response use 2 spaces to represent each indentation level. Do not use the Tab key to indent, as this will not indent and instead move you to the next field.

Instructions

Information

Please answer all questions.

This exam is worth 60% of your overall mark.

To answer a question that r se use 2 spaces to represent each indentation level. Do not use the Tab key to in dent and instead move you to the next field.

Multiple Choice

Question 1

3 Marks

This question is about hash tables with open addressing:

A hash table of size 10 uses hash function **hash(key) = key % 10** and linear probing to handle collisions. After inserting 6 keys into an empty hash table, the corresponding hash table is as follows: [None, None, 32, 13, 54, 22, 46, 53, None, None]

Which one of the following [illegible]e order in which the key values could have been inserted in the table?

Select one:

○ a. 46, 32, 54, 22, 13, 53

○ b. 54, 32, 13, 22, 53, 46

○ c. 46, 54, 32, 13, 22, 53

○ d. 32, 46, 53, 13, 54, 22

○ e. 53, 46, 22, 54, 13, 32

○ f. None of the above

Question 2

3 Marks

Consider the array representation of Heap class as seen in the lectures. Which one of the following arrays does **not** represent a **max heap**.

Select one:

○ a. [None,9,5,6,3,4,1]

○ b. [None,11,9,3,5,6,1]

○ c. [None,8,6,5,3,2,2]

○ d. [None,15,6,10,2,7,8]

○ e. [None,11,7,9,5,6,8]

○ f. None of the above.

MIPS

Information

To answer a question that requires a code response use 2 spaces to represent each indentation level. Do not use the Tab key to indent, as this will not indent and instead move you to the next field.

Question 3

This question is about MIPS [illegible] sheet is included below.)

3
Marks

MIPS reference sheet for FIT1008 and FIT2085

Semester 2, 2019

[illegible] System calls

Call code ($v0)	Service	[illegible]	Returns	Notes
1	Print integer	[illegible]	-	value is signed
4	Print string	[illegible] to print	-	string must be terminated with '\0'
5	Input integer	[illegible]	$v0 = entered integer	value is signed
8	Input string	[illegible] ich the string will be stored $a1 = maximum number of characters in the string	–	returns if $a1-1 characters or Enter typed, the string is terminated with '\0'
9	Allocate memory	$a0 = number of bytes	$v0 = address of first byte	-
10	Exit	-	-	ends simulation

Table 2: General-purpose registers

Number	Name	Purpose
R00	$zero	provides constant zero
R01	$at	reserved for assembler
R02, R03	$v0, $v1	system call code, return value
R04–R07	$a0--$a3	system call and function arguments
R08–R15	$t0--$t7	temporary storage (caller-saved)
R16–R23	$s0--$s7	temporary storage (callee-saved)
R24, R25	$t8, $t9	temporary storage (caller-saved)
R28	$gp	pointer to global area
R29	$sp	stack pointer
R30	$fp	frame pointer
R31	$ra	return address

Table 3: Assembler directives

.data	assemble into data segment
.text	assemble into text (code) segment
.word w1[, w2, ...]	allocate word(s) with initial value(s)
.space n	allocate n bytes of uninitialized, unaligned space
.ascii "string"	allocate ASCII string, do not terminate
.asciiz "string"	allocate ASCII string, terminate with '\0'

Table 4: Function calling convention

On function call:	**Caller:** saves temporary registers on stack passes arguments on stack calls function using `jal fn_label`	**Callee:** saves value of `$ra` on stack saves value of `$fp` on stack copies `$sp` to `$fp` allocates local variables on stack
On function return:	**Callee:** sets `$v0` to return value clears local variables off stack restores saved `$fp` off stack restores saved `$ra` off stack returns to caller with `jr $ra`	**Caller:** clears arguments off stack restores temporary registers off stack uses return value in `$v0`

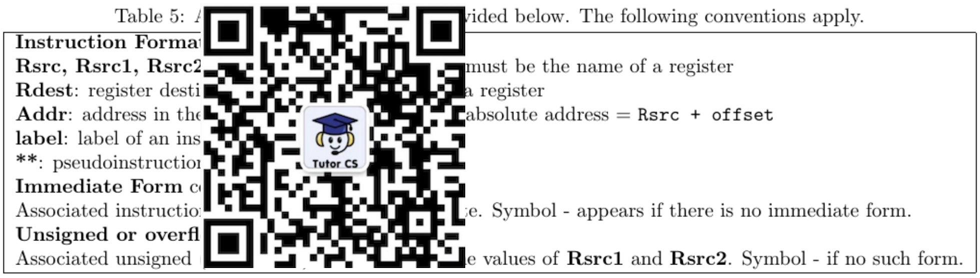


Table 5: A ... vided below. The following conventions apply.

Instruction Forma

Rsrc, Rsrc1, Rsrc2 must be the name of a register

Rdest: register desti a register

Addr: address in the absolute address = `Rsrc + offset`

label: label of an ins

******: pseudoinstruction

Immediate Form c

Associated instructio e. Symbol - appears if there is no immediate form.

Unsigned or overfl

Associated unsigned e values of **Rsrc1** and **Rsrc2**. Symbol - if no such form.

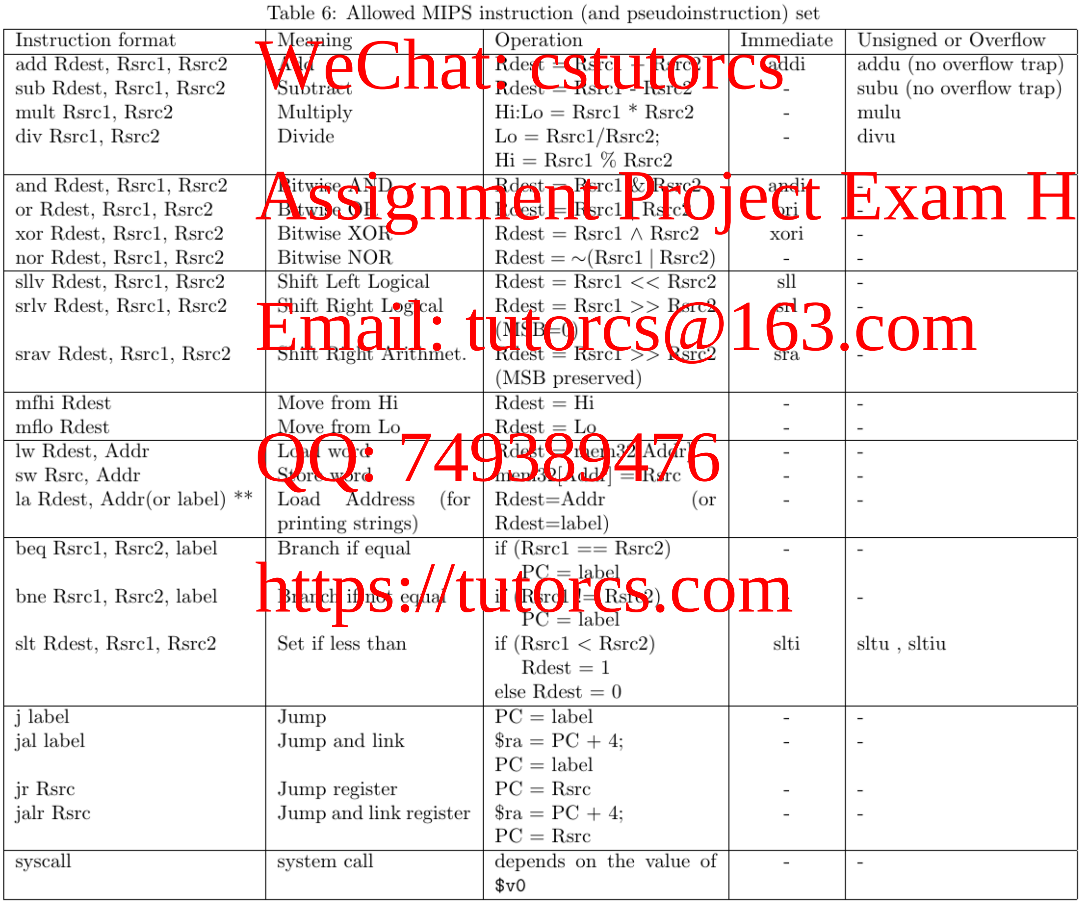


Table 6: Allowed MIPS instruction (and pseudoinstruction) set

Instruction format	Meaning	Operation	Immediate	Unsigned or Overflow
add Rdest, Rsrc1, Rsrc2	Add	Rdest = Rsrc1 + Rsrc2	addi	addu (no overflow trap)
sub Rdest, Rsrc1, Rsrc2	Subtract	Rdest = Rsrc1 - Rsrc2	-	subu (no overflow trap)
mult Rsrc1, Rsrc2	Multiply	Hi:Lo = Rsrc1 * Rsrc2	-	mulu

div Rsrc1, Rsrc2	Divide	Lo = Rsrc1/Rsrc2;
		Hi = Rsrc1 % Rsrc2
			-	divu

and Rdest, Rsrc1, Rsrc2	Bitwise AND	Rdest = Rsrc1 & Rsrc2	andi	-
or Rdest, Rsrc1, Rsrc2	Bitwise OR	Rdest = Rsrc1 | Rsrc2	ori	-
xor Rdest, Rsrc1, Rsrc2	Bitwise XOR	Rdest = Rsrc1 ∧ Rsrc2	xori	-
nor Rdest, Rsrc1, Rsrc2	Bitwise NOR	Rdest = ∼(Rsrc1 | Rsrc2)	-	-
sllv Rdest, Rsrc1, Rsrc2	Shift Left Logical	Rdest = Rsrc1 << Rsrc2	sll	-
srlv Rdest, Rsrc1, Rsrc2	Shift Right Logical	Rdest = Rsrc1 >> Rsrc2 (MSB=0)	srl	-
srav Rdest, Rsrc1, Rsrc2	Shift Right Arithmet.	Rdest = Rsrc1 >> Rsrc2 (MSB preserved)	sra	-
mfhi Rdest	Move from Hi	Rdest = Hi	-	-
mflo Rdest	Move from Lo	Rdest = Lo	-	-
lw Rdest, Addr	Load word	Rdest = mem32[Addr]	-	-
sw Rsrc, Addr	Store word	mem32[Addr] = Rsrc	-	-
la Rdest, Addr(or label) **	Load Address (for printing strings)	Rdest=Addr (or Rdest=label)	-	-

beq Rsrc1, Rsrc2, label	Branch if equal	if (Rsrc1 == Rsrc2)
		PC = label
			-	-

bne Rsrc1, Rsrc2, label	Branch if not equal	if (Rsrc1 != Rsrc2)
		PC = label
			-	-

slt Rdest, Rsrc1, Rsrc2	Set if less than	if (Rsrc1 < Rsrc2)
		Rdest = 1
		else Rdest = 0
			slti	sltu , sltiu

j label	Jump	PC = label	-	-
jal label	Jump and link	$ra = PC + 4; PC = label	-	-
jr Rsrc	Jump register	PC = Rsrc	-	-
jalr Rsrc	Jump and link register	$ra = PC + 4; PC = Rsrc	-	-
syscall	system call	depends on the value of `$v0`	-	-

Assume you want to translate to MIPS the Python condition

```
if x <= y:
```

where both **x** and **y** are global variables. Which of the following pieces of MIPS code correctly translates the Python condition? Indicate what each line of the selected code does, or (if "None of the above") provide the correct code.

a)

```
lw $t0, x
lw $t1, y
slt $t2, $t0, $t1
beq $t2, $0, endif
```

b)

```
lw $t0, x
lw $t1, y
slt $t2, $t0, $t1
bneq $t2, $0, endif
```

c)

```
lw $t0, x
lw $t1, y
slt $t2, $t1, $t0
bne $t2, $0, endif
```

d)

```
lw $t0, x
lw $t1, y
slt $t2, $t1, $t0
beq $t2, $0, endif
```

e)

```
slt $t2, x, y
beq $t2, $0, endif
```

f)

```
slt $t2, x, y
bne $t2, $0, endif
```

g)

```
slt $t2, y, x
beq $t2, $0, endif
```

h)

```
slt $t2, y, x
bne $t2, $0, endif
```

i)

None of the above.

Question 4

This question is about MIPS. (The MIPS reference sheet is included below.)

3 Marks

MIPS reference sheet for FIT1008 and FIT2085

Semester 2, 2019

[illegible] System calls

Call code (`$v0`)	Service	[illegible]	Returns	Notes
1	Print integer	[illegible]	-	value is signed
4	Print string	[illegible] to print	-	string must be terminated with `'\0'`
5	Input integer	[illegible]	`$v0` = entered integer	value is signed
8	Input string	[illegible] hich the string will be stored `$a1` = maximum number of characters in the string	–	returns if `$a1-1` characters or Enter typed, the string is terminated with `'\0'`
9	Allocate memory	`$a0` = number of bytes	`$v0` = address of first byte	-
10	Exit	-	-	ends simulation

Table 2: General-purpose registers

Number	Name	Purpose
R00	`$zero`	provides constant zero
R01	`$at`	reserved for assembler
R02, R03	`$v0, $v1`	system call code, return value
R04–R07	`$a0--$a3`	system call and function arguments
R08–R15	`$t0--$t7`	temporary storage (caller-saved)
R16–R23	`$s0--$s7`	temporary storage (callee-saved)
R24, R25	`$t8, $t9`	temporary storage (caller-saved)
R28	`$gp`	pointer to global area
R29	`$sp`	stack pointer
R30	`$fp`	frame pointer
R31	`$ra`	return address

Table 3: Assembler directives

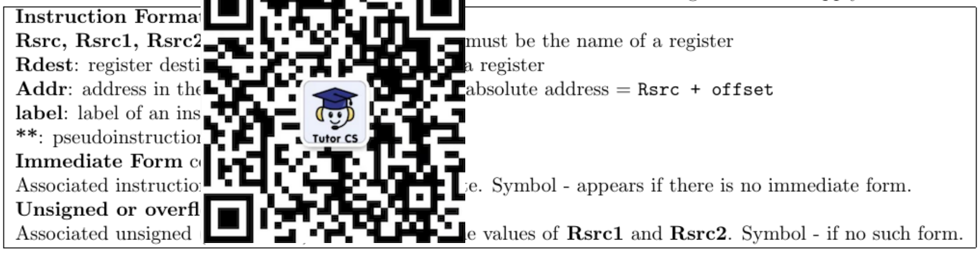
.data	assemble into data segment
.text	assemble into text (code) segment
.word w1[, w2, ...]	allocate word(s) with initial value(s)
.space n	allocate n bytes of uninitialized, unaligned space
.ascii ”string”	allocate ASCII string, do not terminate
.asciiz ”string”	allocate ASCII string, terminate with `'\0'`

Table 4: Function calling convention

	Caller:	**Callee:**
On function call:	saves temporary registers on stack passes arguments on stack calls function using `jal fn_label`	saves value of `$ra` on stack saves value of `$fp` on stack copies `$sp` to `$fp` allocates local variables on stack

	Callee:	**Caller:**
On function return:	sets `$v0` to return value clears local variables off stack restores saved `$fp` off stack restores saved `$ra` off stack returns to caller with `jr $ra`	clears arguments off stack restores temporary registers off stack uses return value in `$v0`

Table 5: A… …vided below. The following conventions apply.



Instruction Forma…

Rsrc, Rsrc1, Rsrc2… must be the name of a register

Rdest: register desti… a register

Addr: address in the… absolute address = `Rsrc + offset`

label: label of an ins…

******: pseudoinstructio…

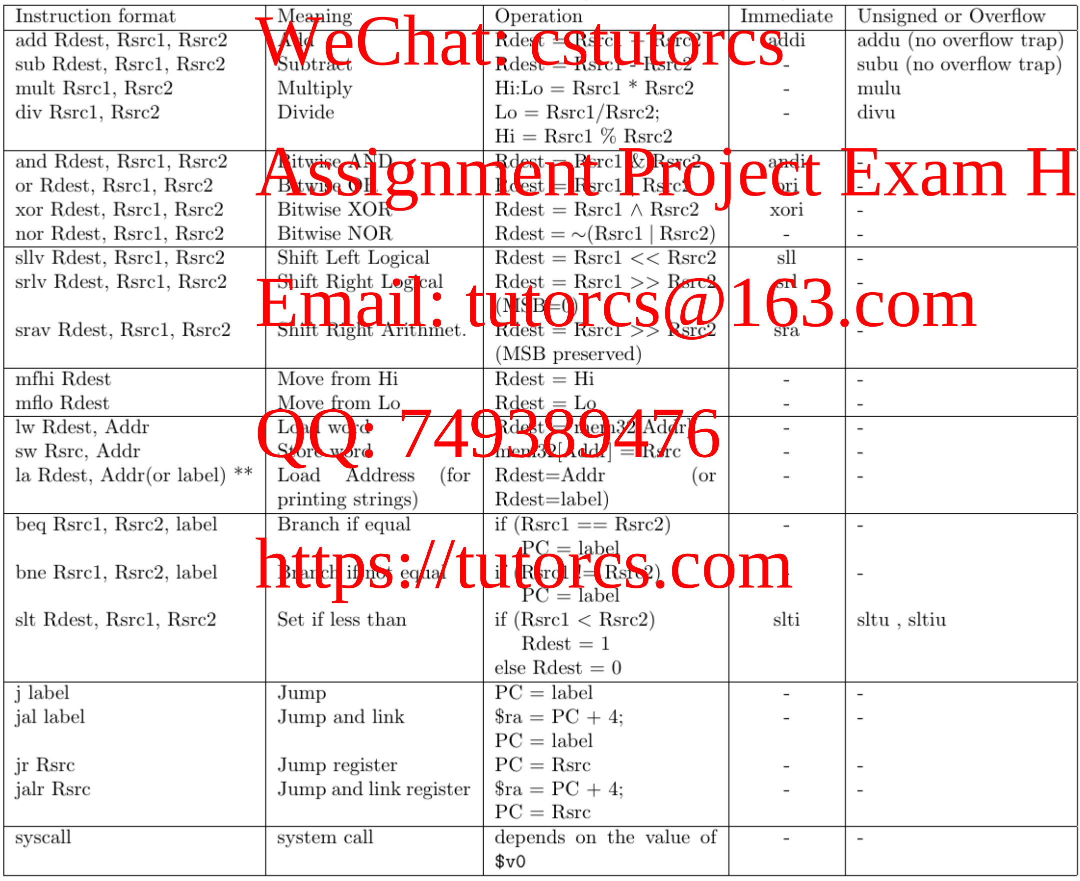
Immediate Form c…

Associated instructio… …e. Symbol - appears if there is no immediate form.

Unsigned or overfl…

Associated unsigned … …e values of **Rsrc1** and **Rsrc2**. Symbol - if no such form.

Table 6: Allowed MIPS instruction (and pseudoinstruction) set



Instruction format	Meaning	Operation	Immediate	Unsigned or Overflow

add Rdest, Rsrc1, Rsrc2
sub Rdest, Rsrc1, Rsrc2
mult Rsrc1, Rsrc2
div Rsrc1, Rsrc2
	Add
	Subtract
	Multiply
	Divide
		Rdest = Rsrc1 + Rsrc2
		Rdest = Rsrc1 - Rsrc2
		Hi:Lo = Rsrc1 * Rsrc2
		Lo = Rsrc1/Rsrc2;
		Hi = Rsrc1 % Rsrc2
			addi
			-
			-
			-
				addu (no overflow trap)
				subu (no overflow trap)
				mulu
				divu

and Rdest, Rsrc1, Rsrc2
or Rdest, Rsrc1, Rsrc2
xor Rdest, Rsrc1, Rsrc2
nor Rdest, Rsrc1, Rsrc2
	Bitwise AND
	Bitwise OR
	Bitwise XOR
	Bitwise NOR
		Rdest = Rsrc1 & Rsrc2
		Rdest = Rsrc1 | Rsrc2
		Rdest = Rsrc1 ∧ Rsrc2
		Rdest = ∼(Rsrc1 | Rsrc2)
			andi
			ori
			xori
			-
				-
				-
				-
				-

sllv Rdest, Rsrc1, Rsrc2
srlv Rdest, Rsrc1, Rsrc2
srav Rdest, Rsrc1, Rsrc2
	Shift Left Logical
	Shift Right Logical
	Shift Right Arithmet.
		Rdest = Rsrc1 << Rsrc2
		Rdest = Rsrc1 >> Rsrc2 (MSB=0)
		Rdest = Rsrc1 >> Rsrc2 (MSB preserved)
			sll
			srl
			sra
				-
				-
				-

mfhi Rdest
mflo Rdest
	Move from Hi
	Move from Lo
		Rdest = Hi
		Rdest = Lo
			-
			-
				-
				-

lw Rdest, Addr
sw Rsrc, Addr
la Rdest, Addr(or label) **
	Load word
	Store word
	Load Address (for printing strings)
		Rdest = mem32[Addr]
		mem32[Addr] = Rsrc
		Rdest=Addr (or Rdest=label)
			-
			-
			-
				-
				-
				-

beq Rsrc1, Rsrc2, label
bne Rsrc1, Rsrc2, label
slt Rdest, Rsrc1, Rsrc2
	Branch if equal
	Branch if not equal
	Set if less than
		if (Rsrc1 == Rsrc2)
		PC = label
		if (Rsrc1 != Rsrc2)
		PC = label
		if (Rsrc1 < Rsrc2)
		Rdest = 1
		else Rdest = 0
			-
			-
			slti
				-
				-
				sltu , sltiu

j label
jal label
jr Rsrc
jalr Rsrc
	Jump
	Jump and link
	Jump register
	Jump and link register
		PC = label
		$ra = PC + 4;
		PC = label
		PC = Rsrc
		$ra = PC + 4;
		PC = Rsrc
			-
			-
			-
			-
				-
				-
				-
				-

syscall	system call	depends on the value of `$v0`	-	-

The following piece of MIPS code translates the Python array access

```
x = the_list[i-1]
```

where **x**, **the_list** and **i** are all global variables. The code is correct if**i>0**. This might seem strange, as the MIPS code does not include an instruction to subtract 1 from index **i**. Explain why this is not needed when **i>0** (no explanation no marks).

```
lw $t0, i
lw $t1, the_list
addi $t2, $0, 4
mult $t0, $t2
mflo $t0
add $t0, $t0, $t1
lw $t0, ($t0)
sw $t0, x
```

Sorting

Information

To answer a question that requires a code response use 2 spaces to represent each indentation level. Do not use the Tab key to indent, as this will not indent and instead move you to the next field.

Question 5

8
Marks

This question is about sorti

Consider the following piec

```
def some_sort(the_array
  n = len(the_array)

  for p in range(n):
    tmp = the_array[p]
    q = 0
    while the_array[q] < tmp:
      q += 1

    while q <= p:
      r = the_array[q]
      the_array[q] = tmp
      tmp = r
      q += 1
```

Is this a correct sorting algorithm? If so, explain the invariant that ensures correctness; if not, give an example where it fails.

Question 6

7
Marks

Given the following implementation of Quick Sort, implement the partition function, choosing the *last* element of the list as pivot.

```
def quick_sort(array):
    start = 0
    end = len(array) - 1
    quick_sort_aux(array, start, end)

def quick_sort_aux(array, start, end):
    if start < end:
        boundary = partition(array, start, end)
        quick_sort_aux(array, start, boundary - 1)
        quick_sort_aux(array, boundary + 1, end)
```

Data Structures

Question 7

4

Marks

This question is about Data Structures.

Consider a **List** class implemented using arrays, whose partial implementation is as follows:

```
    def __init__(self, size):
        if size <= 0:
            raise Value                 be positive")
        self.array = [N
        self.count = 0

    def size(self):
        return self.cou

    def is_empty(self):
        return self.size() == 0

    def is_full(self):
        return self.size() >= len(self.array)
```

Add to the class an implementation of the method

```
__contains__(self, item)
```

which returns **True** if **item** appears in the list. Note that __contains__ should have O(N) worst case complexity (where n is the number of items in the list).

Question 8

7

Marks

Consider a **List** class implemented using arrays, whose partial implementation is as follows:

```
    def __init__(self, size):
        if size <= 0:
            raise ValueError("Size should be positive")
        self.array = [None] * size
        self.count = 0

    def size(self):
        return self.count

    def is_empty(self):
        return self.size() == 0

    def is_full(self):
        return self.size() >= len(self.array)
```

Add also to the **List** class an implementation of the method

```
remove_first(self)
```

which removes and returns the element at index 0, ensuring all other elements are correctly swapped to the left. You should appropriately account for any errors.

Question 9

6

Marks

Consider the following partial implementation of a **Queue** class, which is implemented using only two **Stacks** (with the usual operations).

```
class Queue:
    def __init__(self):
        self.left = Stack()
        self.right = Stack()
```

Explain how you would impl **item)** and **serve(self)** methods with this representation. Try to make ble (you do not need to give code).

Linked Lists

Question 10

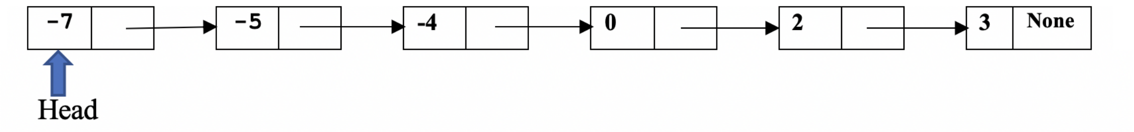
8
Marks

This question is about understanding code that uses linked lists. Consider a List class implemented using linked nodes, and defined as follows:

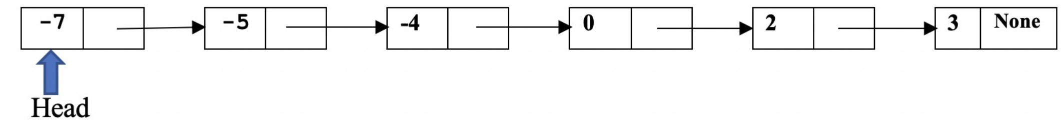
```
class Node:
    def __init__(self, item, link = None):
        self.item = it
       self.link = link

class List:
    def __init__(self):
        self.head = Non

    def add(self, item):
        self.head = Node(item,self.head)

    def mystery(self):
        while (self.head is not None and self.head.item < 0):
          self.head = self.head.link
        if self.head is not None:
            previous = self.head
            current = self.head.link
           while current is not None:
               if current.item < 0:
                   previous.link = current.link
                else:
                    previous = current
                current = current.link
```

Which list of those shown below is the result of calling themystery method for linked list?

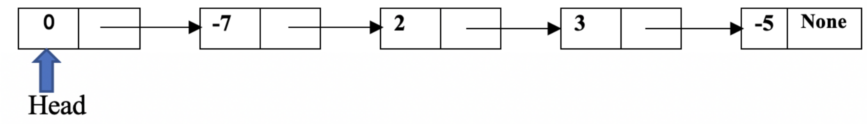
Option A:



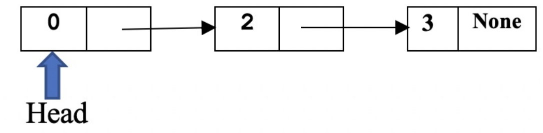
Option B:



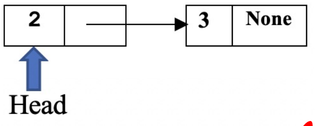
Option C:



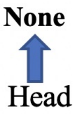
Option D:



Option E:



Option F:



Option G:

None of the above

Explain why (no explanation [illegible] ver is "None of the above" provide the elements of the correct resulting list.

Recursion

Question 11

2
Marks

This question is about Recursion.

You are visiting a friend, who has previously completed FIT1008/FIT2085, for the weekend. But when you arrive, your friend is nowhere to be found; and instead of their door code, they have just left you a pair of numbers, and the fol

```
def clue(x, y):
  if y == 0:
    return 1
  elif y % 2 == 1:
    y = clue(x, y-1)
    return x * y
  else:
    y = clue(x, y//2)
    return y * y
```

You may assume all inputs are **positive integers**.

Write the result of the function **clue** for the input values:

- x = 100, y = 1

Question 12

2
Marks

This question is about Recursion.

You are visiting a friend, who has previously completed FIT1008/FIT2085, for the weekend. But when you arrive, your friend is nowhere to be found; and instead of their door code, they have just left you a pair of numbers, and the following clue:

```
def clue(x, y):
  if y == 0:
    return 1
  elif y % 2 == 1:
    y = clue(x, y-1)
    return x * y
  else:
    y = clue(x, y//2)
    return y * y
```

You may assume all inputs are **positive integers**.

Write the result of the function **clue** for the input values:

- x = 4, y = 2

Question 13

2

Marks

This question is about Recursion.

You are visiting a friend, who has previously completed FIT1008/FIT2085, for the weekend. But when you arrive, your friend is nowhere to be found; and instead of their door code, they have just left you a pair of numbers, and the following clue:

```
def clue(x, y):
  if y == 0:
    return 1
  elif y % 2 == 1:
    y = clue(x, y-1)
    return x * y
  else:
    y = clue(x, y//2)
    return y * y
```

You may assume all inputs are **positive integers.**

Write the result of the function **clue** for the input values:

- x = 2, y = 3

Question 14

2

Marks

This question is about Recursion.

You are visiting a friend, who has previously completed FIT1008/FIT2085, for the weekend. But when you arrive, your friend is nowhere to be found; and instead of their door code, they have just left you a pair of numbers, and the following clue:

```
def clue(x, y):
  if y == 0:
    return 1
  elif y % 2 == 1:
    y = clue(x, y-1)
    return x * y
  else:
    y = clue(x, y//2)
    return y * y
```

You may assume all inputs are **positive integers.**

Write the result of the function **clue** for the input values:

- x = 2, y = 5

Question 15

3

Marks

You are visiting a friend, who has previously completed FIT1008/FIT2085, for the weekend. But when you arrive, your friend is nowhere to be found; and instead of their door code, they have just left you a pair of numbers, and the following clue:

```
def clue(x, y):
  if y == 0:
    return 1
  elif y % 2 == 1:
    y = clue(x, y-1)
    return x * y
  else:
    y = clue(x, y//2)
    return y * y
```

You may assume all inputs are positive integers.

What is the worst-case time complexity of clue (using big-O time complexity)? Explain your answer (no explanation, no marks).

Binary Search Trees

Question 16

9 Marks

This question is about Binary Search Trees.

Complete the missing expressions (#1 to #5) of **sum_leq**, which recursively computes the sum of all elements in a BST tree which are less than or equal to **item**.

```
def sum_leq(self, it[illegible]
    return self.sum_[illegible]item)

def sum_leq_aux(self[illegible]
    if tree is None:
        return #1
    else:
        low_sum = #2
        if tree.item == item:
            return #3
        elif #4
          return low_sum
        else:
          return #5
```

Question 17

2 Marks

The following method either returns the sum of the items in the tree that are ancestors of the node whose item is **value** in the tree, or returns **None** if **value** is not present.

```
def sum_parents(node, value):
    if node is None:
        return None
    elif node.item == value:
        return 0
    elif value < node.item:
        below = sum_parents(node.left, value)
    else:
        below = sum_parents(node.right, value)
    if below is None:
        return None
    else:
        return below + node.item
```

What is the worst-case complexity of **sum_parents**? (Remember to define any variables you use). For what kind of trees does this occur?

Question 18

Complete the tail-recursive function **sum_parents_tail**, which is computes the same value as **sum_parents**

5 Marks

```
def sum_parents_tail(node, value):
      return sum_parents_tail_aux(node, value, 0)

  def sum_parents_tail_aux(node, value, acc):
      if node is None:
          return #1
      elif node.item ==
          return #2
      else:
          #3
```

Heaps

Question 19

This question is about heaps.

8
Marks

Consider an implementation of a **MaxHeap**, which provides the following methods:

- **__init__(self)**, which creates a **MaxHeap** object
- **add(self, item)**, which add
- **get_max(self)**, which rem highest value item from the Heap.

Write a function **find_kth_sm** nds the **kth smallest** item in array **alist** (with 1 being the smallest). You must sta with the following code:

```
def find_kth_smallest(a
    mx = MaxHeap()
    n = len(alist)
    # TODO
```

You cannot use any additional MaxHeaps or other data structures.

Assertions / Exceptions

Question 20

4

Marks

This question is about Assertions and Exceptions.

Consider the following (somewhat strange) code:

```
class MyError(Exception):
    pass

def mystery1(a, b):
    assert a >= b, "a s[illegible] equal to b"
    if a != b:
        raise MyError("[illegible]")
    return 15

def mystery2(a, b):
    try:
        return mystery1(a, b)
    except MyError:
        return 8

if __name__ == '__main__':
    try:
        result = mystery2(3, 4)
        print("Result:" + str(result))
    except AssertionError as e:
        print(e)
    except MyError as e:
        print(e)
```

Select the correct output, and explain your selection (no explanation, no marks).

- It will terminate normally and output "a is not equal to b".
- It will terminate normally and output "a should be greater or equal to b".
- It will terminate normally and output "Result: 15".
- It will terminate normally and output "Result: 8".
- It will terminate normally, but produces no output.
- It will crash with an Assertion Error.
- It will crash with a Type Error.
- It will crash with MyError "a is less than b".