MONASH
University

Semester One 2019
Examination Period

Faculty of Information Technology

EXAM CODES:	FIT1008-FIT2085
TITLE OF PAPER:	Introduction to Computer Science
EXAM DURATION:	3 hours 10 mins

Rules

<u>Authorised Materials</u>

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

if yes, items permitted are:

Instructions

Please answer all questions online. Noting and calculations to be done in the scriptbook or working sheets provided.

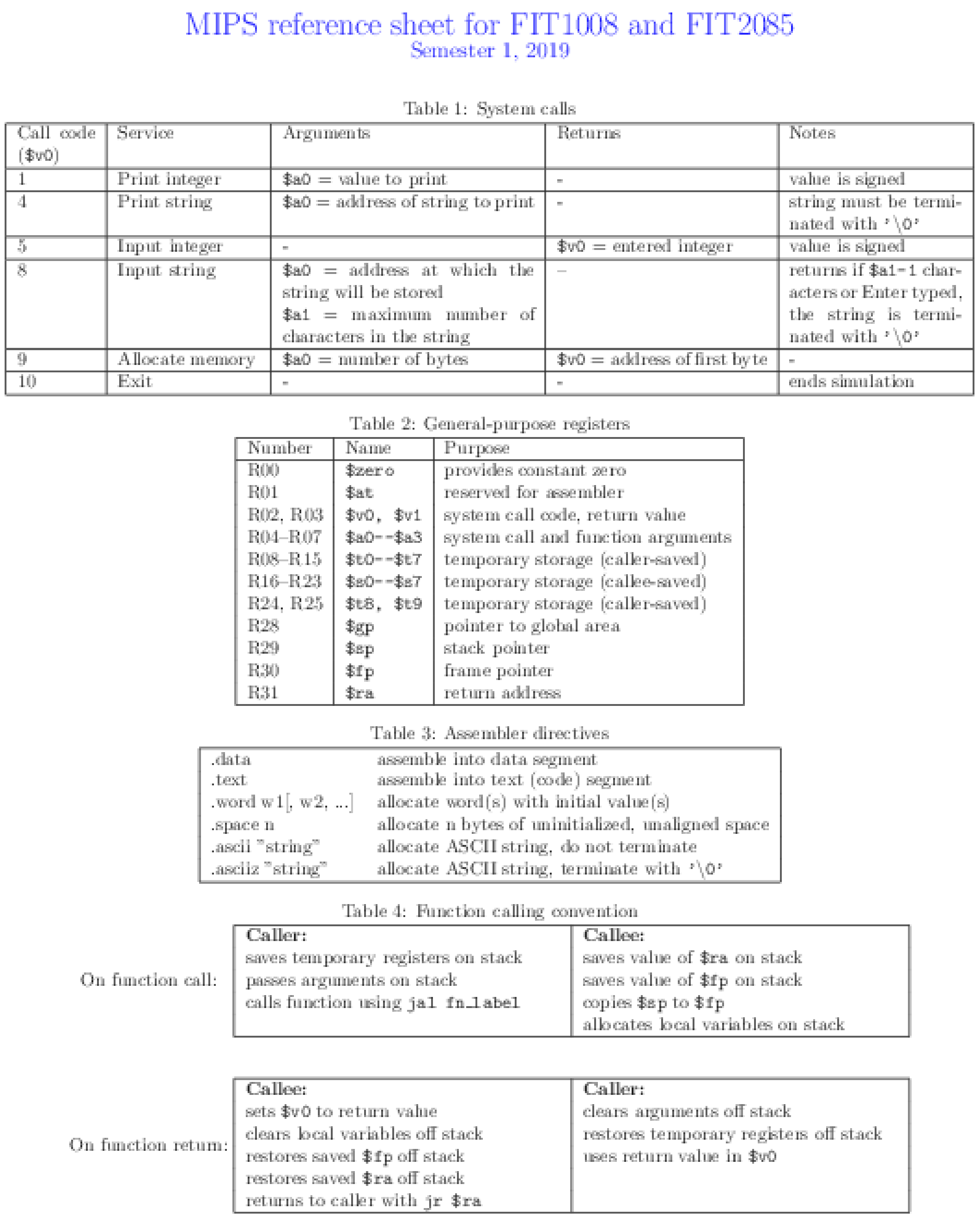
Instructions

Information

Please answer all questions online. Noting and calculations to be done in the scriptbook or working sheets provided.

MIPS reference sheet

Information



MIPS reference sheet for FIT1008 and FIT2085
Semester 1, 2019

Table 1: System calls

Call code ($v0)	Service	Arguments	Returns	Notes
1	Print integer	$a0 = value to print	-	value is signed
4	Print string	$a0 = address of string to print	-	string must be terminated with '\0'
5	Input integer	-	$v0 = entered integer	value is signed
8	Input string	$a0 = address at which 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: Allowed MIPS instruction (and pseudoinstruction) set

Instruction format	Meaning	Operation	Immediate	Unsigned format
add Rdest, Rsrc1, Src2	Add	Rdest = Rsrc1 + Src2	addi	addu (no overflow trap)
sub Rdest, Rsrc1, Src2	Subtract	Rdest = Rsrc1 - Src2	-	subu (no overflow trap)
mult Rsrc1, Src2	Multiply	Hi:Lo = Rsrc1 * Src2	-	mulu
div Rsrc1, Src2	Divide	Lo = Rsrc1/Src2; Hi = Rsrc1 % Src2	-	divu
and Rdest, Rsrc1, Src2	Bitwise AND	Rdest = Rsrc1 & Src2	andi	-
or Rdest, Rsrc1, Src2	Bitwise OR	Rdest = Rsrc1 | Src2	ori	-
xor Rdest, Rsrc1, Src2	Bitwise XOR	Rdest = Rsrc1 ∧ Src2	xori	-
nor Rdest, Rsrc1, Src2	Bitwise NOR	Rdest = ∼(Rsrc1 | Src2)	-	-
sllv Rdest, Rsrc1, Src2	Shift Left Logical	Rdest = Rsrc1 << Src2	sll	-
srlv Rdest, Rsrc1, Src2	Shift Right Logical	Rdest = Rsrc1 >> Src2 (MSB=0)	srl	-
srav Rdest, Rsrc1, Src2	Shift Right Arithmetic	Rdest = Rsrc1 >> Src2 (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, Src2	Set if less than	if (Rsrc1 < Src2)
		Rdest = 1
		else Rdest = 0
			slti	sltu

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	-	-

Python to MIPS translation

Question 1

17
Marks

Translate the following Python code faithfully into MIPS assembly language. Make sure you follow the MIPS function calling and memory usage conventions.

```
def func (n):
    if n <= 0:
        result = 0
    else:
        result = 4*n+func(n-1)
    return result
```

We ask that you translate this code*instruction by instruction*, using the sub-questions below. The concatenation of all 6 answers must correspond to the entire translation of the Python code above.

Comments are not mandatory and will not be marked, but will help us understand your code and, in that way, can help you get extra points. Use '#' at the start of a line in your MIPS code to add a comment.

```
def func (n):
```

```
    if n <= 0:
```

```
        result = 0
```

```
    else:
```

```
        result = 4*n+func(n-1)
```

```
    return result
```

The Python function above can easily be implemented iteratively, rather than recursively. Assume the iterative version uses **N** bytes of **Heap** memory. What value is **N** and how many bytes will the recursive version use? Explain why (no explanation no marks).

Assume now that the iterative version uses **N** bytes of **Stack** memory. What value is **N** and how many bytes will the recursive version use? Explain why (no explanation no marks).

Scoping with classes in Python

Question 2

10
Marks

You are provided with the Python *myclass.py* module:

```python
class myclass:
    def __init__(self,x):
        self.x = x

    def a(self):
        self.x = self.x + 1

    def b(self):
        self.x = x + 2

    def c(self):
        x = self.x + 3

    def __str__(self):
        return str(self.x)

def a(x):
    x = x - 1

def b():
    x = x + 2
```

This module will be imported in each of the following questions. In each of the following questions, there will be exactly one print statement. We ask that you find the value being printed by each piece of code.

For example, if the question is:

```python
print(1)
```

Then your answer should be: 1.

```python
from myclass import *
myobject = myclass(1)
print(myobject)
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
x = 2
myobject = myclass(x)
x = 1
print(myobject)
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
myclass.x = 3
myobject = myclass(2)
print(myobject)
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
myobject = myclass(3)
myclass.x = 4
print(myobject.x)
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
myclass.x = 6
myobject = myclass(myclass.x)
a(myclass.x)
print(myobject.x)
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
x = 5
myclass.x = 3
print(myclass(1).b())
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
x = 7
myobject = myclass(x)
myobject.c()
print(x)
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
x = 6
b()
myobject = myclass(x)
myobject.a()
print(myobject)
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
x = 2
print(myclass(a(x)))
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

```python
from myclass import *
x = 0
myobject = myclass(x)
yourobject =
myclass(myobject.x)
myobject.c()
a(yourobject.x)
print(yourobject)
```

· 5 · 3 · 9 · 10 · No output. The code produces an error. · 2 · 7 · 4 · 1 · None · 6 · 0 · 8

ADT and linear structures

Question 3

18
Marks

In this exercise we will create two classes. The first class, called**Node**, provides an Abstract Data Type with methods:

- get_data(self) returns the data contained at the Node.
- get_next(self) returns the next Node, None if there is none.
- get_previous(self) returns the previous Node, None if there is none.
- set_data(self, newdata) sets the data contained at the Node.
- set_next(self, newnext) sets the next Node.

Note that any class Node studied this semester may or may not satisfy the requirements of this question.

The second class is called **Deque**. It is a Double-Ended Queue that is implemented using linked nodes, each of the type Node defined above, It provides an ADT with the following methods (and worst-case time complexities):

- size(self) returns the number of elements in the Deque, in O(1) operations.
- add_front(self, data) adds a new Node with the given data at the front of the Deque, in O(1) operations.
- add_rear(self, data) adds a new Node with the given data at the rear of the Deque, in O(1) operations.
- remove_front(self) removes the Node at the front of the Deque and returns the corresponding data, in O(1) operations. Assume the Deque is non empty.
- remove_rear(self) removes the Node at the rear of the Deque and returns the corresponding data, in O(1) operations. Assume the Deque is non empty.

Note that you are *not* allowed to use built-in Python data structures such as lists.

We ask you to write these methods, together with an __init__() method for each class, as answers to the questions below. The concatenation of these answers must constitute a functional implementation of the Node and Deque ADTs provided above. No need to provide documentation, comments, assertions or exceptions (these will not be marked).

Define the class Node (with all its methods).

```
class Node:
  def __init__(self,initdata):




  def get_data(self):



  def get_next(self):



  def get_previous(self):



  def set_data(self,newdata):



  def set_next(self,newnext):
```

Define the class Deque below (with all its methods).

```
class Deque:
  def __init__(self):






  def size(self):






  def remove_front(self):







  def remove_rear(self):
```

CS saves the world

Question 4

20 Marks

Year 2069. A malevolent alien species who speaks in Python has sent the following message to Earth:

```python
def mystery(x):
    y = x % 2
    x = x // 2
    if x > 0:
        y = y + mystery(x)
    return y

def enigma(x):
    y = mystery(x)
    if y > 1:
        y = y + enigma(y)
    return y

#puny humans must print:
print(enigma(4095))
```

If we do not compute and send them the result of enigma(4095) within the next 3 hours (plus 10 minutes of reading), the aliens have promised that they would destroy the planet. As Earth's leading computer scientist, the task of saving humanity naturally lies with you. Your former esteemed FIT1008&2085 lecturers, recently retired, have advised you to first answer a series of questions before attempting to compute the final result.

Write the output of the function mystery for the input values:

- x=1
- x=2
- x=3
- x=7
- x=8
- x=15

What does the function mystery compute?

What is the time complexity of mystery, using the O() notation? Prove your answer.

Write the output of the function enigma for the input values:

- x = 1
- x = 2
- x = 3
- x = 7
- x = 8
- x = 15

What does the function enigma compute?

What is the time complexity of enigma, using the O() notation? Prove your answer.

What does enigma(4095) return? Justify your answer.

Natural merging

Question 5

14
Marks

In this question we suppose that all sorting is done in a non-decreasing way.

We propose to write a new sorting algorithm that first detects when part of the data is already sorted. For example, if the input list is [0, 4, 1, 2, 8, 5, 7, 9, 3, 6], then the algorithm will first detect the consecutive items that are already sorted, namely [0, 4], [1,2,8], [5,7,9] and [3,6].

After this, the algorithm will merge these sublists two by two until the entire list is sorted.

Write a function find_intervals which, given a list as input, returns the list of indices between which the input list is already sorted. For our example list, the output would be [0, 2, 5, 8, 10], since the list is sorted between indices 0 and 1, 2 and 4, 5 and 7, 8 and 9. Note that the last index in the output list (10) points outside of the input list.

We ask that you use this template:

```
def find_intervals(list):
    separators = [0]
    #TODO your code here
    return separators
```

In the code above, the variable separators refers to the list of indices that you must return. You may use any Python built-in method for this function.

```
def find_intervals(l):
    separators = [0]
    #TODO your code here




    return separators
```

What is the worst-case time complexity of the find_intervals function you have written? Explain your answer.

Write a function natural_merge which takes the list to sort as an input and sorts it. This function must call the previous function to determine intervals which are already sorted. You will not be penalised if you have not attempted or succeeded the previous questions. For this question you will be marked as if the previous questions had been answered correctly.
The function natural_merge must implement the following algorithm:

1. Find the intervals of the input list where the data is already sorted by calling the previous function.
2. Iterate through the list and merge the first and the second interval together. After the merge, these two intervals become a single interval, hence there is one fewer interval in the list. This continues until there is only a single interval left. For example, for our input list, we would obtain the following steps:
   [0, 4] [1, 2, 8] [5, 7, 9] [3, 6] and interval list [0, 2, 5, 8, 10]
   [0, 1, 2, 4, 8] [5, 7, 9] [3, 6] and interval list [0, 5, 8, 10]
   [0, 1, 2, 4, 5, 7, 8, 9] [3, 6] and interval list [0, 8, 10]
   [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] and interval list [0, 10]

   To do the merging, you must call the function merge provided below:

```python
def merge(l, start, mid, end):
    """Merges the two sorted sublists of l
       between start and mid (excluded)
       and mid and end (excluded) """
    tmp = [None] * (end-start)
    k1 = start
    k2 = mid
    use1 = False
    for k in range(start, end):
        if k1 >= mid:
            use1 = False
        elif k2 >= end:
            use1 = True
        else:
            use1 = (l[k1] <= l[k2])
        if use1 is True:
            tmp[k] = l[k1]
            k1 += 1
        else:
            tmp[k] = l[k2]
            k2 += 1
    for k in range(start, end):
        l[k] = tmp[k]
```

Although we provide the code, you only need to call the function merge according to its documentation. Write the function natural_merge below:

```
def natural_merge(l):
```

What is the worst-case time complexity of the merge function we have provided? Explain your answer.

What is the best-case time complexity of the algorithm natural_merge? Explain your answer. You may answer this question based on the description we provide of it, even if you have not implemented it.

What is the worst-case time complexity of the algorithm natural_merge? Explain your answer. You may answer this question based on the description we provide of it, even if you have not implemented it.

How could a sorting algorithm with better time complexity be designed using the ideas presented in this question? Explain your answers.

Two coin problems
Question 6

8
Marks

We are are presented with a row of n coins with positive or negative values, and we want to pick up the largest amount of money from the row of coins subject to one constraint. We will consider two different constraints, leading to two different variants of this problem.

In the first variant, the constraint is that we cannot pick up any two adjacent coins. For instance, for the input [7, -2, 10, -12, 5], the maximum value that can be picked up is 22, by picking up coin 7, 10 and 5.

In the second variant, the constraint is that we cannot pick up a coin without picking at least one adjacent coin. For instance, for the input [7, -2, 10, -12, 5], the maximum value that can be picked up is 15, by picking up coin 7, -2 and 10. Hence we can pick up two or more coins in a row (for instance 7, -2, 10), but not a single coin (e.g. 5) by itself.

Write a function non_adjacent_coins which takes as input a list of coin values and outputs the *maximum value* of a valid subset of coins (i.e. such that no two adjacent coins are selected). You must start your solution with the following code:

```
def non_adjacent_coins(l):
    n = len(l)
    mem = [None] * (n+1)
```

```
def non_adjacent_coins(l):
  n = len(l)
  mem = [None] * (n+1)
```

Write a function adjacent_coins which takes as input a list of coin values and outputs the*maximum value* of a valid subset of coins (i.e. such that any selected coin is adjacent to at least another selected coin). We will solve this problem by computing the subproblems f(i,j) = "the maximum value achievable among the first i coins (including i) where if j=0, coin i is not selected, if j = 1, coin i is selected, but not coin i -1 (if it exists), and if j=2, coins i and i-1 are both selected", for i=0,...,n and j=0,1 or 2. In the code below we compute and store the value f(i,j) in three arrays, one for each possible value of j.

You must use the following code as a template, where we have marked with #TODO the four missing parts of the code that you must write:

```
def adjacent_coins(l):
    n = len(l)
    #We compute the max achievable with the first i coins
    #in three different cases:
    zero_taken = [None] * (n+1)#if i is not picked
    one_taken = [None] * (n+1)#if i is picked but not i-1
    two_plus_taken = [None] * (n+1)#if i and i-1 are picked
    #we initialise the arrays
    zero_taken[0] = 0
    one_taken[0] = - math.inf
    two_plus_taken[0] = - math.inf
    #we iterate to compute all other values
    for i in range(1, n+1):
        zero_taken[i] = #TODO
        one_taken[i] = #TODO
        two_plus_taken[i] = #TODO
    #we return the maximum value
    return #TODO
```

Note that math.inf returns infinity. Write your answer below:

```
import math

def adjacent_coins(l):
    n = len(l)
    #We compute the max achievable with the first i coins
    #in three different cases:
    zero_taken = [None] * (n+1)#if i is not picked
    one_taken = [None] * (n+1)#if i is picked but not i-1
    two_plus_taken = [None] * (n+1)#if i and i-1 are picked
    #we initialise the arrays
    zero_taken[0] = 0
    one_taken[0] = - math.inf
    two_plus_taken[0] = - math.inf
    #we iterate to compute all other values
    for i in range(1, n+1):
```

k smallest elements

Question 7

9
Marks

Given an unordered list with n elements that can be compared in constant time, and might contain duplicates, we want to find the k smallest elements.
Explain how the k smallest elements can be found in time O(n log n) and O(n) auxiliary memory.

Explain how the k smallest elements can be found in time O(kn) and O(1) auxiliary memory.

Explain how the k smallest elements can be found in time O(n + k log n) and O(n) auxiliary memory.

Explain how the k smallest elements can be found in time O(n log k) and O(k) auxiliary memory.

Resolving collisions

Question 8

2
Marks

Suppose you are given the following set of keys to insert into a Hash table of size 11:

22 , 23 , 37, 29, 33, 39, 2, 27, 21, 17.

The hash function is given below.

```
def hash_function(key):
    return key%11
```

Suppose that we use linear probing with a skip of k=1 to resolve collisions. Select the correct state of the Hash Table after the keys have been inserted.

Select one:

- ○ 22, 23, 33, 2, 37, 27, 39, 29, 17, None, 21
- ○ 22, 23, 37, 29, 33, 39, 2, 27, 21, 17
- ○ 29, 37, 17, 33, 21, 2, 39, 23, 22, None, 27
- ○ 39, 2, 22, 17, 23, 37, 29, None, 33, 21, 27
- ○ 2, 37, None, 17, 23, 29, 39, 27, 33, 21, 22
- ○ None, 2, 23, 21, 39, 33, 29, 22, 27, 37, 17
- ○ 22, 23, 37, 29, 33, None, 27, 39, 17, 21, 2
- ○ 17, 21, None, 2, 29, 23, 27, 22, 39, 33, 37
- ○ 37, 2, 21, 29, 22, 33, None, 23, 17, 27, 39
- ○ 21, 2, 22, 37, 39, 17, 23, 29, 33, 27, None
- ○ 33, 29, 27, 37, 2, 22, 17, 21, None, 23, 39
- ○ There is no correct state. Not all keys can be inserted.
- ○ 22, 23, 2, None, 37, 27, 39, 29, 17, 33, 21
- ○ 22, 23, 2, 17, 37, 27, 39, 29, None, 33, 21
- ○ 22, 23, 2, 33, 37, 27, 39, 29, 17, None, 21
- ○ 22, 23, None, 2, 37, 27, 39, 29, 17, 33, 21
- ○ 22, 23, 2, None, 37, 27, 39, 29, 17, 33, 21

Question 9

2 Marks

Suppose you are given the following set of keys to insert into a Hash table of size 11:

22 , 23 , 37, 29, 33, 39, 2, 27, 21, 17.

The hash function is given below.

```
def hash_function(key):
    return key%11
```

Suppose that we use quadratic probing to resolve collisions. Select the correct state of the Hash Table after the keys have been inserted.

Select one:

- ○ 29, 37, 17, 33, 21, 2, 39, 23, 22, None, 27
- ○ 39, 2, 22, 17, 23, 37, 29, None, 33, 21, 27
- ○ 2, 37, None, 17, 23, 29, 39, 27, 33, 21, 22
- ○ None, 2, 23, 21, 39, 33, 29, 22, 27, 37, 17
- ○ 17, 21, None, 2, 29, 23, 27, 22, 39, 33, 37
- ○ 37, 2, 21, 29, 22, 33, None, 23, 17, 27, 39
- ○ 29, 23, 17, 22, 39, 2, 33, 21, None, 27, 37
- ○ 33, 29, 27, 37, 2, 22, 17, 21, None, 23, 39
- ○ There is no correct state. Not all keys can be inserted.
- ○ 22, 23, 2, None, 37, 27, 39, 29, 17, 33, 21
- ○ 22, 23, 33, 2, 37, 27, 39, 29, 17, None, 21
- ○ 22, 23, 2, 33, 37, 27, 39, 29, 17, None, 21
- ○ 22, 23, 2, 33, 37, 27, 39, 29, None, 17, 21
- ○ 22, 23, 2, 17, 37, 27, 39, 29, 33, None, 21
- ○ 22, 23, 2, None, 37, 27, 17, 29, 33, 39, 21