NAMIBIA UNIVERSITY OF SCIENCE AND TECHNOLOGY

## FACULTY OF COMPUTING AND INFORMATICS

## DEPARTMENT OF COMPUTER SCIENCE

| | |
|---|---|
| QUALIFICATION: BACHELOR OF COMPUTER SCIENCE HONOURS | |
| QUALIFCATION CODE: 08BCSH | LEVEL: 8 |
| COURSE: FORMAL METHODS | COURSE CODE: FMM810S |
| DATE: January 2020 | SESSION: 2 |
| DURATION: 3 Hours | MARKS: 80 |

| | |
|---|---|
| SECOND OPPORTUNITY/SUPPLEMENTARY EXAMINATION QUESTION PAPER | |
| EXAMINER: | Prof. José G. Quenum |
| MODERATOR: | Dr Gokop Longinus Goteng |

This paper consists of 2 pages
(excluding this front page)

INSTRUCTIONS

1. This paper contains 4 questions.
2. Answer all questions on the exam paper.
3. Marks/scores are provided at the right end of each question
4. Do not use or bring into the examination venue books, programmable calculators, mobile devices and other materiaks that may provide you with unfair advantage. Should you be in possession of one right now, draw the attention of the examiner officer or the invigilator.
5. NUST examination rules and regulations apply.

PERMISSIBLE MATERIALS

None

## Question 1 ..... [17 points]

Consider a complex system with a multitude of processes. Inside the system there is a closed group where processes which have joined the group can share messages from the outside world. All processes part of the system form the procs set. Processes which have joined the group are part of the joined set, while processes which have not joined yet or have left are part of the left set.

(a) Assuming a type Process for processes, define the CompS schema representing the complex system. [2]

(b) Define the schema of the initSys operation that initialises the complex system. [2]

(c) Define the schema of the join operation that adds a process p to the group. [3]

(d) Define the schema of the leave operation that removes a process p from the group. [3]

(e) Define the schema of the create operation that adds a process p to the complex system. [2]

(f) Define the schema of the query operation that checks if a process p is part of the group. In case the process has not been created yet, the query operation has to indicate so. [5]

## Question 2 ..... [25 points]

We introduce Danz an abstract data store with a finite number of values (of type V). The state of a Danz object includes:

contents: a sequence of values;

size: the number of values currently in the sequence;

max_size: the contents capacity.

We introduce three (03) operations to manipulate Danz:

initialise to initialise the data store;

insert to insert new values into the contents of the data store;

fetch to retrieve data from the contents of the data store.

Note that after instantiation the capacity of the data store cannot change.

(a) Using the schema notation of the Z specification formalism, propose a specification for Danz [3]

(b) Specify all three operations defined for Danz [8]

(c) In order to provide the user with a proper notification, we introduce a free type, OutCome, which has three values: op_ok, contents_full and contents_empty. Using the new type we can return op_ok when an operation is successful. We can also notify the user with one of the other values in case an insert or a fetch operation fails. Define the free type and using a schema specify the successful notification. [3]

(d) Define the error handling versions of insert and fetch [8]

(e) Define the preconditions for initialise, insert and fetch [3]

## Question 3 ..... [15 points]

Consider a relation W : $\{\alpha, \gamma, \tau, \pi\} \leftrightarrow \{k, u, m, r\}$ as follows: $\{\alpha \mapsto u, \gamma \mapsto k, \tau \mapsto m\}$

(a) Give in extension the following sets: dom(W) and ran(W) [2]

(b) Give in extension the following sets: $\{\tau\} \lhd$ W and W $\rhd \{r\}$ [4]

(c) Give in extension the following sets: $\{\pi\} \lhd\!\!\!= $ W and W $\rhd\!\!\!= \{k, m\}$ [4]

(d) What is the inverse of W? [2]

(e) What is the reflexive closure for W? [3]

## Question 4 ..... [23 points]

Given a set RecVal, consider a type RecT where each element is either of type SimpleRT or CompoRT. An element of type SimpleRT holds a value of type RecVal, while an element of type CompoRT holds a pair of values each one of type RecT.

(a) Using free types in Z, formally define RecT [6]

(b) Define a function $f_1$ that reverses the content of a sequence of values [4]

(c) Next we introduce two functions $f_2$ and $f_3$. $f_2$ behaves as follows; applied to an element of type RecT, when it is a SimpleRT it returns a sequence containing the only element held by the type. On the other hand, when the element is of type CompoRT, it returns a concatenation of the image of the first component and the image of the second component (in the indicated order). As for $f_3$, when applied to a SimpleRT it returns the same value, while applied to a CompoRT it returns another CompoRT with the component's position switched. Define $f_2$ and $f_3$. [8]

(d) Prove that the sequential composition $\fatsemi$ of $f_3$ and $f_2$ is equivalent to that of $f_2$ and $f_1$ [5]