COMP SCI 2ME3 and SFWR ENG 2AA4 Midterm Examination
McMaster University

DAY CLASS Dr. S. Smith
DURATION OF EXAMINATION: 3 hours
MCMASTER UNIVERSITY MIDTERM EXAMINATION March 4, 2021

NAME: [Hriday Jham —SS]

Student ID: [400227516 —SS]

This examination paper includes 17 pages and 4 questions. You are responsible for ensuring that your copy of the examination paper is complete. Bring any discrepancy to the attention of your instructor.

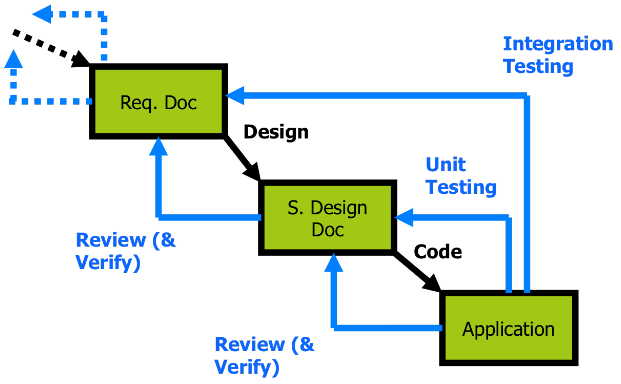
By submitting this work, I certify that the work represents solely my own independent efforts. I confirm that I am expected to exhibit honesty and use ethical behaviour in all aspects of the learning process. I confirm that it is my responsibility to understand what constitutes academic dishonesty under the Academic Integrity Policy.

Special Instructions:

1. For taking tests remotely:
   - Turn off all unnecessary programs, especially Netflix, YouTube, games like Xbox or PS4, anything that might be downloading or streaming.
   - If your house is shared, ask others to refrain from doing those activities during the test.
   - If you can, connect to the internet via a wired connection.
   - Move close to the Wi-Fi hub in your house.
   - Restart your computer, 1-2 hours before the exam. A restart can be very helpful for several computer hiccups.
   - Commit and push your tex file, compiled pdf file, and code files frequently.
   - Ensure that you push your solution (tex file, pdf file and code files) before time expires on the test. The solution that is in the repo at the deadline is the solution that will be graded.
2. It is your responsibility to ensure that the answer sheet is properly completed. Your examination result depends upon proper attention to the instructions.
3. All physical external resources are permitted, including textbooks, calculators, computers, compilers, and the internet.
4. The work has to be completed individually. Discussion with others is strictly prohibited.

5. Read each question carefully.
6. Try to allocate your time sensibly and divide it appropriately between the questions.
7. The set $\mathbb{N}$ is assumed to include 0.

Question 1 [6 marks] Parnas advocates faking a rational design process as depicted in the figure below. The faked documentation follows these steps: Requirements (SRS) → Design (MG and MIS) → Application Implementation (code) → Verification and Validation (Unit Testing, Integration Testing, Review). How are the principles of a) abstraction and b) separation of concerns applied in a rational design process? In your answer you can refer to any aspects of the process, documentation, and/or Parnas's principles.



[Fill in your answer below —SS]

a) Abstraction

- Since Abstraction is the principle of focusing on what is important and ignoring the irrelevant, it is implemented while faking a rational design process as we are focusing at one part of the software design process at a time and only focusing on what is important for that part of the design process.
- In a rational design process, we postpone details until implementation which is a principle of abstraction as stated by parnas.
- While working on a design, we implement only the parts we understand and those that are the components required for the document to give the desired output while constantly understanding and fixing what we didn't understand when we started. This further enforces the implementation of only the important details while leaving irrelevant details out.

b) Separation of Concerns

- Seperation of concerns is the principle that different concerns should be isolated and considered seperately. The goal is to reduce complex problems into simpler problems, which is exactly what we do while implementing a rational design process. We start with the required document, then we design a solution,implement the code and finally test and verify the code.
- Seperation of concerns enables parallelization. In rational design, we do not fully understand the problem at first and work through the software while constantly reverting back to previous stages and fixing errors, thus using parallelization
- By implementing abstraction, we implement seperation of concerns as Abstraction is a special case of seperation of concerns.

Consider the specification for two modules: SeqServices and SetOfInt.

Sequence Services Library

Module

SeqServicesLibrary

Uses

None

Syntax

Exported Constants

None

Exported Types

None

Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
max_val	seq of $\mathbb{Z}$	$\mathbb{N}$	ValueError
count	$\mathbb{Z}$, seq of $\mathbb{Z}$	$\mathbb{N}$	ValueError
spices	seq of $\mathbb{Z}$	seq of string	ValueError
new_max_val	seq of $\mathbb{Z}$, $\mathbb{Z} \rightarrow \mathbb{B}$	$\mathbb{N}$	ValueError

Semantics

State Variables

None

State Invariant

None

Assumptions

- All access programs will have inputs provided that match the types given in the specification.

Access Routine Semantics

max_val(s)

- output: $out := |m| : \mathbb{N}$ such that $(m \in s) \land \forall(x : \mathbb{Z} | x \in s : |m| \geq |x|)$
- exception: $(|s| = 0 \Rightarrow \text{ValueError})$

count(t, s)

- output: $out := +(x : \mathbb{Z} | x \in s \land x = t : 1)$
- exception: $(|s| = 0 \Rightarrow \text{ValueError})$

spices(s)

- output: $out := \langle x : \mathbb{Z} | x \in s : (x \leq 0 \Rightarrow \text{“nutmeg”} | \text{True} \Rightarrow \text{“ginger”}) \rangle$
- exception: $(|s| = 0 \Rightarrow \text{ValueError})$

new_max_val($s,\ f$)

- output: $out := \text{max\_val}(\langle x : \mathbb{Z} | x \in s \land f(x) : x \rangle)$
- exception: $(|s| = 0 \Rightarrow \text{ValueError})$

Set of Integers Abstract Data Type

Template Module

SetOfInt

Uses

None

Syntax

Exported Types

SetOfInt = ?

Exported Constants

None

Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
new SetOfInt	seq of $\mathbb{Z}$	SetOfInt	
is_member	$\mathbb{Z}$	$\mathbb{B}$	
to_seq		seq of $\mathbb{Z}$	
union	SetOfInt	SetOfInt	
diff	SetOfInt	SetOfInt	
size		$\mathbb{N}$	
empty		$\mathbb{B}$	
equals	SetOfInt	$\mathbb{B}$	

Semantics

State Variables

s: set of $\mathbb{Z}$

State Invariant

None

Assumptions

- The SetOfInt constructor is called for each object instance before any other access routine is called for that object. The constructor can only be called once. All access programs will have inputs provided that match the types given in the specification.

Access Routine Semantics

new SetOfInt(x_s):

- transition: $s := \cup(x : \mathbb{Z} | x \in x_s : \{x\})$
- output: $out := self$
- exception: none

is_member(x):

- output: $x \in s$
- exception: none

to_seq():

- output: $out :=$ set_to_seq(s)
- exception: none

union(t):

- output: SetOfInt(set_to_seq(s)$||t$.to_seq())
  # in case it is clearer, an alternate version of output is:
  SetOfInt(set_to_seq($s \cup \{x : \mathbb{Z} | x \in t$.to_seq() $: x\}$))
- exception: none

diff(t):

- output: SetOfInt(set_to_seq($s \cap$ complement(t.to_seq())))
- exception: none

size():

- output: $|s|$
- exception: none

empty():

- output: $s = \varnothing$
- exception: none

equals(t):

- output: $\forall(x : \mathbb{Z} | x \in \mathbb{Z} : x \in t.\text{to\_seq}() \leftrightarrow x \in s)$ *# this means:* t.to_seq() $= s$
- exception: none

Local Functions

set_to_seq : set of $\mathbb{Z} \rightarrow$ seq of $\mathbb{Z}$
set_to_seq$(s) \equiv \langle x : \mathbb{Z} | x \in s : x \rangle$ # *Return a seq of all of the elems in the set s, order does not matter*

complement : seq of $\mathbb{Z} \rightarrow$ set of $\mathbb{Z}$
complement$(A) \equiv \{x : \mathbb{Z} | x \notin A : x\}$

Question 2 [15 marks]
[Complete Python code to match the above specification. —SS] The files you need to complete are: `SeqServicesLibrary.py` and `SetOfInt.py`. Two testing files are also provided: `expt.py` and `test_driver.py`. The file `expt.py` is pre-populated with some simple experiments to help you see the interface in use, and do some initial test. You are free to add to this file to experiment with your work, but the file itself isn't graded. The `test_driver.py` is also not graded. However, you may want to create test cases to improve your confidence in your solution. The stubs of the necessary files are already available in your `src` folder. The code will automatically be imported into this document when the `tex` file is compiled. You should use the provided Makefile to test your code. You will NOT need to modify the Makefile. The given Makefile will work for `make test`, without errors, from the initial state of your repo. The `make expt` rule will also work, because all lines of code have been commented out. Uncomment lines as you complete work on each part of the modules relevant to those lines in `expt.py` file. The required imports are already given in the code. You should not make any modifications in the provided import statements. You should not delete the ones that are already there. Although you can solve the problem without adding any imports, if your solution requires additional imports, you can add them. As usual, the final test is whether the code runs on mills.
Any exceptions in the specification have names identical to the expected Python exceptions; your code should use exactly the exception names as given in the spec.
You do not need to worry about doxygen comments. However, you should include regular comments in the code where it would benefit from an explanation.
You do not need to worry about PEP8. Adherence to PEP8 will not be part of the grading.
Remember, your code needs to implement the given specification so that the interface behaves as specified. This does NOT mean that the local functions need to all be implemented, or that the types used internally to the spec need to be implemented exactly as given. If you do implement any local functions, please make them private by preceding the name with double underscores.

Code for SeqServicesLibrary.py

```
## @file SeqServicesLibrary.py
#  @author Hriday Jham
#  @brief Library module that provides functions for working with
   sequences
#  @details This library assumes that all functions will be provided
   with arguments of the expected types
#  @date 03/04/2021

#returning the maximum asbolute value
def max_val(s):
    if len(s) == 0:
        raise ValueError("Invalid Length of input")
    else :
        maximum = 0
        for i in range(len(s)):
            if abs(s[i]) > abs(maximum):
                maximum = s[i]
        return abs(maximum)

#returning the occurences of t in the Sequence s
def count(t, s):
    if len(s) == 0:
        raise ValueError("Invalid Length of input")
    else :
        counter = 0
        for i in range (len(s)):
            if s[i] == t:
                counter += 1
        return counter

#Returns a customized Sequence
def spices(s):
    if len(s) == 0:
        raise ValueError("Invalid Length of input")
    else :
        output = []
        for i in s:
            if i <= 0:
                output.append("nutmeg")
            else:
                output.append("ginger")
        return output

#Returns the maximum absolute value after applying f to the sequence
```

```
def new_max_val(s, f):
    if len(s) == 0:
        raise ValueError("Invalid Length of input")
    else :
        new_list = []
        for x in s:
            if f(x) == True:
                new_list.append(x)
        return max_val(new_list)
```

Code for SetOfInt.py

```
## @file SetOfInt.py
#  @author Hriday Jham
#  @brief Set of integers
#  @date 03/04/2021


class SetOfInt:

    #Constructor for SetOfInt
    def __init__(self, xs):
        set1 = []
        for x in xs:
            set1.append(x)
        self.s = set(set1)

    #returns whether x is a member of the set
    def is_member(self, x):
        return x in self.s

    #returns the set as a sequence
    def to_seq(self):
        return list(self.s)

    #returns the union of the set with the set taken as parameter
    def union(self, t):
        self_list = self.to_seq()
        t_list = t.to_seq()
        for i in t_list:
            self_list.append(i)
        return SetOfInt(self_list)

    #Returns the disjunction of complement. i.e. Integers which are in
    #one set but not the other
    def diff(self, t):
        self_list = self.to_seq()
        t_list = t.to_seq()
        new_list = []
        for i in self_list:
            if i not in t_list:
                new_list.append(i)
        for x in t_list:
            if x not in self_list:
                new_list.append(x)
        return SetOfInt(new_list)
    #returns size of set
    def size(self):
        return len(self.s)

    #returns whether set is empty
    def empty(self):
        return self.size() <= 0

    #checks whether two sets are equal
    def equals(self, t):
        flag = True
        for i in self.s:
            if i in t.to_seq() == False:
                flag = False
        return flag
```

Code for expt.py

```
## @file expt.py
#  @author Spencer Smith
#  @brief This file is intended to help test that your interface
   matches the specified interface
#  @date 03/04/2021

from SeqServicesLibrary import *
from SetOfInt import *

# Exercising Sequence Services Library
print()
print("SeqServicesLibrary, max_val expt:", max_val([1, 2, -3]))
print("SeqServicesLibrary, count expt:", count(1, [1, 1, 1, 1]))
print("SeqServicesLibrary, spices expt:", spices([-5, 0, 23]))
print("SeqServicesLibrary, new_max_val expt:", new_max_val([-5, 0,
   23], lambda x: x >10))
print()

# Exercising Set of Integers
xs = [-9, 6, 23, 21, -5]
ys = list(xs)
ys.append(99)
S = SetOfInt(xs)
print("SetOfInt, is_member expt:", S.is_member(21))
print("SetOfInt, to_seq expt:", S.to_seq())
S2 = SetOfInt(ys)
S3 = S.union(S2)
print("SetOfInt, union expt:", S3.to_seq())
S4 = S2.diff(S)
print("SetOfInt, diff expt:", S4.to_seq())
print("SetOfInt, size expt:", S4.size())
print("SetOfInt, size expt:", S4.empty())
S5 = SetOfInt([-9, 6, 23, -5, 21])
print("SetOfInt, equals expt:", S.equals(S5))
print()
```

Code for test_driver.py

```
## @file test_driver.py
#  @author Your Name
#  @brief Tests implementation of SeqServicesLibrary and SetOfInt ADT
#  @date 03/04/2021

from SeqServicesLibrary import *
from SetOfInt import *

from pytest import *

## @brief Tests functions from SeqServicesLibrary.py
class TestSeqServices:

    # Sample test
    def test_sample_test1(self):
        assert True

## @brief Tests functions from SetOfInt.py
class TestSetOfInt:

    # Sample test
    def test_sample_test2(self):
        assert True
```

Question 3 [5 marks]
Critique the design of the interface for the SetOfInt module. Specifically, review the interface with respect to its consistency, essentiality, generality and minimality. Please be specific in your answer.
[Put your answer for each quality below. —SS]

- **consistency**: The SetOfInt module is not consistent because it does not follow uniform name conventions. For example, if the Boolean functions were being named is_member, then empty and equals shouldve been named is_empty and is_equals.
- **essentiality**: SetOfInt is not essential as empty is being reproduced by using size. Both of these methods were not necessary as having one wouldve sufficed.
- **generality**: The module is general as it is only specific to integers and not generic. It cannot be used in multiple situations for different purposes.
- **minimality**: It is not minimal as the method diff is calculating the complement as well as the disjunction of the set with the complement. Thus the method diff does two jobs instead of one.

Question 4 [4 marks]
The module SetOfInt is for a set of integers. Please answer the following questions related to making that module generic.

a. How would you change the specification to make it generic? (Specifically what changes would you make to the given specification. You don't need to redo the spec, just summarize what changes you would need to make.)

b. What changes would you need to make to the Python implementation to make it generic for type T? (Again, you can describe and characterize the changes; you don't actually have to make them.)

c. What relational operator needs to be defined for type T to be a valid choice?

d. BONUS (1 mark) How would you specify (in the MIS) the relational operator constraint (from the previous question) on the generic type T?

[Put your answer below. —SS]

a. To change the specification of SetOfInt and make it generic, instead of taking sequences of $\mathbb{Z}$ as input, I would take sequences of T for the methods where T is a generic Data Type. Also I would rename the template module and name it SetOfInt

b. Since python does not require dynamic typing, and is already a generic language, I would simply rename SetOfInt to SetOfT.

c. the __eq__ (equal) relational operator must be defined for type T to be valid.

d. (BONUS) T must satisfy __eq__