QUESTION 1

Let’s use a function named isPhoneNumber() to check whether a string matches this pattern, returning either True or False.

Regular expressions, called regexes for short, are descriptions for a pattern of text. For example, a \d in a regex stands for a digit character—that is, any single numeral 0 to 9. The regex \d\d\d-\d\d\d- \d\d\d\d is used by Python to match the text containing phone numbers in the isPhoneNumber() function: a string of three numbers, a hyphen, three more numbers, another hyphen, and four numbers. Any other string would not match the \d\d\d-\d\d\d-\d\d \d\d regex. But regular expressions can be much more sophisticated. For example, adding a 3 in curly brackets ({3}) after a pattern is like saying, “Match this pattern three times.” So the slightly shorter regex \d{3}-\d{3}-\d{4} also matches the correct phone number format.

All the regex functions in Python are in the re module. Enter the following into an interactive shell to import this module:

```
import re


def isPhoneNumber(text):
  phone_pattern = r'[0-9]{3}-\d\d\d-[0-9]{4}'
  return bool(re.search(phone_pattern, text))
```

Example usage:

```
# "123-456-7890" matches a phone number: True
print("\"123-456-7890\" matches a phone number: "+ str(isPhoneNumber("123-456-7890")))
```

```
"123-456-7890" matches a phone number: True
```

```
# "abc-456-7890" matches a phone number: False
print("\"abc-456-7890\" matches a phone number: "+str(isPhoneNumber("abc-456-789
```

```
"abc-456-7890" matches a phone number: False
```

```
# "Here's a number: 123-456-7999" matches a phone number: True
print("\"Here's a number: 123-456-7999\" matches a phone number: "+str(isPhoneNum
```

```
"Here's a number: 123-456-7999" matches a phone number: True
```

Python code: Entire program

```
import re

def isPhoneNumber(text):
    phone_pattern = r'[0-9]{3}-\d{3}-\d{4}'
    return bool(re.search(phone_pattern, text))

if __name__ == "__main__":
    print(str(isPhoneNumber("123-456-7890")))  # TRUE
    print(str(isPhoneNumber("This has a number 450-213-4221")))  # TRUE
    print(str(isPhoneNumber("123-421-542a")))  # FALSE
```

```
True
True
False
```

QUESTION 2

Unix-based operating systems usually include a tool named tail. It displays the last 10 lines of a file whose name is provided as a command line argument.

Write a program that provides the same behavior i.e. the user can specify how many lines to print from the terminal. Display an appropriate error message if the file requested by the user does not exist, or if the command line argument is omitted.

Extend this program to display the: (a) first n lines of a file i.e. head and (b) the middle n lines of a file i.e. middle.

Python code: Tail, head, middle functions

```
import os

def tail(filename, n):
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()
            if n > len(lines):
                n = len(lines)
            for line in lines[-n:]:
                print(line, end='')
    except FileNotFoundError:
        print(f"Error: The file '{filename}' does not exist.")

def head(filename, n):
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()
            if n > len(lines):
                n = len(lines)
            for line in lines[:n]:
                print(line, end='')
    except FileNotFoundError:
        print(f"Error: The file '{filename}' does not exist.")

def middle(filename, n):
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()
            total_lines = len(lines)
            if n > total_lines:
                n = total_lines
            start_index = (total_lines - n) // 2
            for line in lines[start_index:start_index + n]:
                print(line, end='')
    except FileNotFoundError:
        print(f"Error: The file '{filename}' does not exist.")
```

Example usage of functions using example helper method with sample.txt

```
def example(filename,mode,n):
  try:
      n = int(n)
      mode = mode.lower()
      if mode == 'tail':
          tail(filename, n)
      elif mode == 'head':
          head(filename, n)
      elif mode == 'middle':
          middle(filename, n)
      else:
          print("Error: Invalid mode.")
```

```
    except ValueError:
        print("Error: n must be an integer.")
```

Creating a sample text file

```
sample_text = """This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
This is line 6
This is line 7
This is line 8
This is line 9
This is line 10
This is line 11
This is line 12
This is line 13
This is line 14
This is line 15"""

with open('sample.txt', 'w') as file:
    file.write(sample_text)
```

Middle 6 elements

```
example('sample.txt','middle',6);
```

Show hidden output

```
This is line 5
This is line 6
This is line 7
This is line 8
This is line 9
This is line 10
```

Last 3 elements

```
example('sample.txt','tail',3)
```

Show hidden output

```
This is line 13
This is line 14
```

This is line 15

First 2 elements

```
example('sample.txt','head',2)
```

Show hidden output

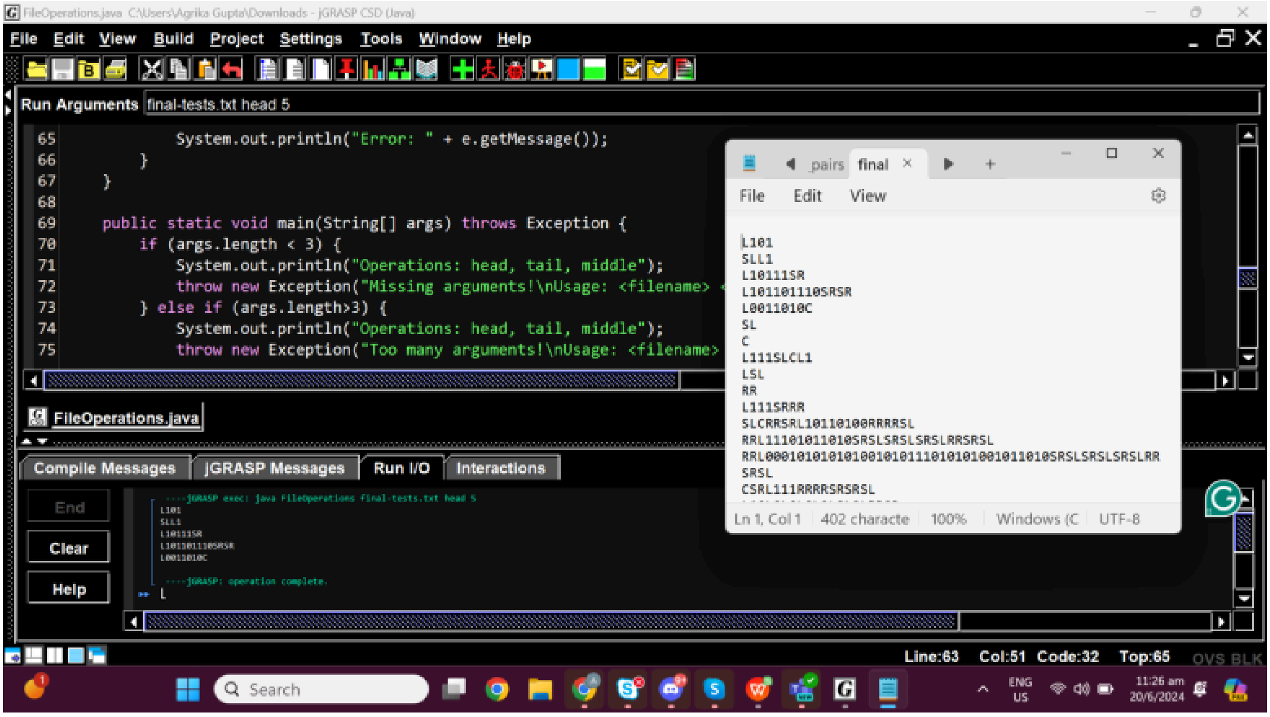
This is line 1
This is line 2

Java code: Entire program

```
import java.io.*;
import java.util.*;

public class FileOperations {
// Function to read the first n lines (head)
public static void head(String filename, int n) {
    try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
        String line;
        int count = 0;
        while ((line = br.readLine()) != null && count < n) {
            System.out.println(line);
            count++;
        }
    } catch (FileNotFoundException e1) {
        System.out.println("File not found!");
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

// Function to read the last n lines (tail)
public static void tail(String filename, int n) {
    try {
        List<String> lines = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
            String line;
            while ((line = br.readLine()) != null) {
                lines.add(line);
                if (lines.size() > n) {
                    lines.remove(0);
                }
            }
        }
        for (String line : lines) {
            System.out.println(line);
        }
    } catch (FileNotFoundException e1) {
        System.out.println("File not found!");
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

// Function to read the middle n lines
public static void middle(String filename, int n) {
    try {
        List<String> lines = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
            String line;
            while ((line = br.readLine()) != null) {
                lines.add(line);
            }
        }
        int totalLines = lines.size();
        if (totalLines < n) {
            n = totalLines;
        }
        int startIndex = (totalLines - n) / 2;
        for (int i = startIndex; i < startIndex + n; i++) {
            System.out.println(lines.get(i));
        }
    } catch (FileNotFoundException e1) {
        System.out.println("File not found!");
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

public static void main(String[] args) throws Exception {
    if (args.length < 3) {
        System.out.println("Operations: head, tail, middle");
        throw new Exception("Missing arguments!\nUsage: <filename> <operation> <number of lines>"
    } else if (args.length>3) {
        System.out.println("Operations: head, tail, middle");
        throw new Exception("Too many arguments!\nUsage: <filename> <operation> <number of lines>
    String filename = args[0];
    String operation = args[1];
    int n = Integer.parseInt(args[2]);

    switch (operation) {
        case "head":
            head(filename, n);
            break;
        case "tail":
            tail(filename, n);
            break;
        case "middle":
            middle(filename, n);
            break;
        default:
            System.out.println("Invalid operation. Use head, tail, or middle.");
    }
}
}
```

Testing head function

```
import requests
from PIL import Image
from io import BytesIO
import matplotlib.pyplot as plt

image_url = 'https://github.com/agrikatheprogrammer/VITON-HD/blob/main/Screenshot%20(499
response = requests.get(image_url)
img = Image.open(BytesIO(response.content))

plt.figure(figsize=(10, 10))
plt.imshow(img)
plt.axis('off')  # Hide the axes
plt.show()
```


FileOperations.java C:\Users\Agrika Gupta\Downloads - jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
pairs
final
File Edit View
Search
Line:63 Col:51 Code:32 Top:65
OVS BLK
ENG US
11:26 am 20/6/2024

Run Arguments final-tests.txt head 5

```
                System.out.println("Error: " + e.getMessage());
            }
        }

        public static void main(String[] args) throws Exception {
            if (args.length < 3) {
                System.out.println("Operations: head, tail, middle");
                throw new Exception("Missing arguments!\nUsage: <filename> <
            } else if (args.length>3) {
                System.out.println("Operations: head, tail, middle");
                throw new Exception("Too many arguments!\nUsage: <filename>
```

FileOperations.java

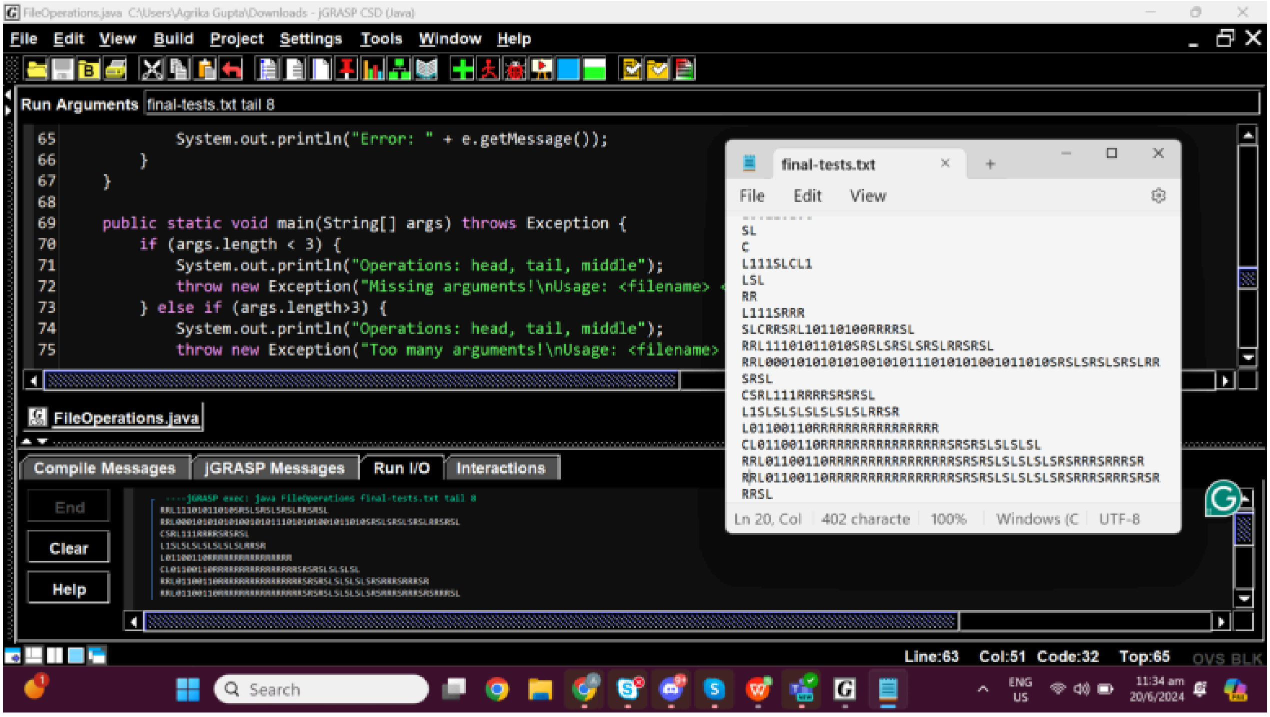
```
L101
SLL1
L10111SR
L101101110SRSR
L0011010C
SL
C
L111SLCL1
LSL
RR
L111SRRR
SLCRRSRL10110100RRRRSL
RRL11101011010SRSLSRSLSRSLRRSRSL
RRL0001010101010010101110101010010110100SRSLSRSLSRSLRR
SRSL
CSRL111RRRRSRSRSL
```

Ln 1, Col 1 | 402 characte | 100% | Windows (C | UTF-8

```
 ----jGRASP exec: java FileOperations final-tests.txt head 5
L101
SLL1
L10111SR
L101101110SRSR
L0011010C

 ----jGRASP: operation complete.
```

Testing tail function

```
import requests
from PIL import Image
from io import BytesIO
import matplotlib.pyplot as plt

image_url = 'https://github.com/agrikatheprogrammer/VITON-HD/blob/main/Screenshot%20(500
response = requests.get(image_url)
img = Image.open(BytesIO(response.content))

plt.figure(figsize=(10, 10))
plt.imshow(img)
plt.axis('off')  # Hide the axes
plt.show()
```



FileOperations.java C:\Users\Agrika Gupta\Downloads - jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help
Run Arguments final-tests.txt tail 8
FileOperations.java
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
Line:63 Col:51 Code:32 Top:65 OVS BLK
Search
ENG US
11:34 am 20/6/2024
final-tests.txt
File Edit View
Ln 20, Col 402 characte 100% Windows (C UTF-8

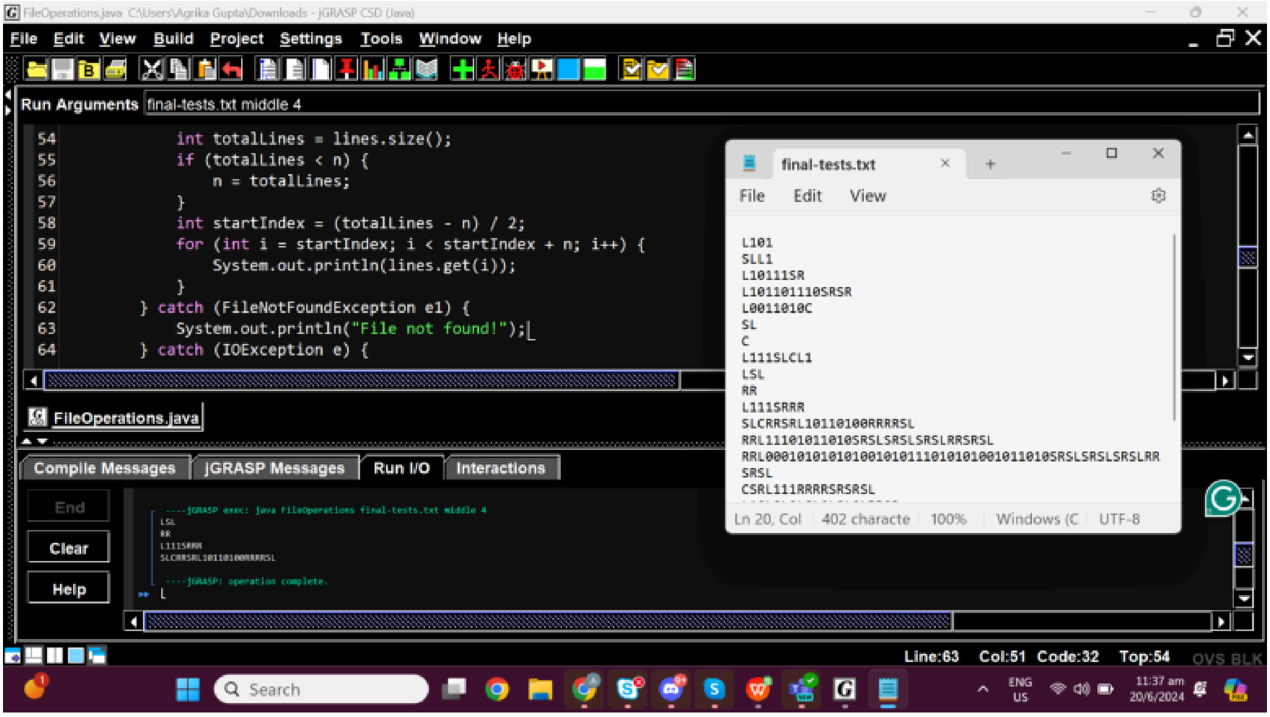
```
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void main(String[] args) throws Exception {
        if (args.length < 3) {
            System.out.println("Operations: head, tail, middle");
            throw new Exception("Missing arguments!\nUsage: <filename>
        } else if (args.length>3) {
            System.out.println("Operations: head, tail, middle");
            throw new Exception("Too many arguments!\nUsage: <filename>
```

```
SL
C
L111SLCL1
LSL
RR
L111SRRR
SLCRRSRL10110100RRRRSL
RRL11101011010SRSLSRSLSRSLRRSRSL
RRL0001010101010010101110101010010110 10SRSLSRSLSRSLRR
SRSL
CSRL111RRRRSRSRSL
L1SLSLSLSLSLSLSLRRSR
L01100110RRRRRRRRRRRRRRRR
CL01100110RRRRRRRRRRRRRRRRRRSRSRSLSLSLSL
RRL01100110RRRRRRRRRRRRRRRRRRSRSRSLSLSLSLSRSRRRSRRRSR
RRL01100110RRRRRRRRRRRRRRRRRRSRSRSLSLSLSLSRSRRRSRRRSRSR
RRSL
```

```
 ----jGRASP exec: java FileOperations final-tests.txt tail 8
RRL11101011010SRSLSRSLSRSLRRSRSL
RRL000101010101001010111010101001011010SRSLSRSLSRSLRRSRSL
CSRL111RRRRSRSRSL
L1SLSLSLSLSLSLSLRRSR
L01100110RRRRRRRRRRRRRRRR
CL01100110RRRRRRRRRRRRRRRRRRSRSRSLSLSLSL
RRL01100110RRRRRRRRRRRRRRRRRRSRSRSLSLSLSLSRSRRRSRRRSR
RRL01100110RRRRRRRRRRRRRRRRRRSRSRSLSLSLSLSRSRRRSRRRSRSRRRSL
```

Testing middle function

```
import requests
from PIL import Image
from io import BytesIO
import matplotlib.pyplot as plt

image_url = 'https://github.com/agrikatheprogrammer/VITON-HD/blob/main/Screenshot%20(501
response = requests.get(image_url)
img = Image.open(BytesIO(response.content))

plt.figure(figsize=(10, 10))
plt.imshow(img)
plt.axis('off')  # Hide the axes
plt.show()
```


FileOperations.java C:\Users\Agrika Gupta\Downloads - jGRASP CSD (Java)
File Edit View Build Project Settings Tools Window Help
FileOperations.java
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
Line:63 Col:51 Code:32 Top:54
final-tests.txt
File Edit View
Ln 20, Col 402 characte 100% Windows (C UTF-8
Search
ENG US
11:37 am 20/6/2024

Run Arguments final-tests.txt middle 4

```
            int totalLines = lines.size();
            if (totalLines < n) {
                n = totalLines;
            }
            int startIndex = (totalLines - n) / 2;
            for (int i = startIndex; i < startIndex + n; i++) {
                System.out.println(lines.get(i));
            }
        } catch (FileNotFoundException e1) {
            System.out.println("File not found!");
        } catch (IOException e) {
```

```
L101
SLL1
L10111SR
L101101110SRSR
L0011010C
SL
C
L111SLCL1
LSL
RR
L111SRRR
SLCRRSRL10110100RRRRSL
RRL111010110101SRSLSRSLSRSLRRSRSL
RRL000101010101001010111010101001011010SRSLSRSLSRSLRR
SRSL
CSRL111RRRRSRSRSL
```

```
----jGRASP exec: java FileOperations final-tests.txt middle 4
LSL
RR
L111SRRR
SLCRRSRL10110100RRRRSL

----jGRASP: operation complete.
```

QUESTION 3

Write a function called SpOdd. The function takes an array of integers as input and return an array of integers with all odd numbers in original array. Keep the same order as the original array.

Python code

```
def SpOdd(arr):
    return [x for x in arr if x % 2 != 0] #functional programming
```

Test function

```
print(SpOdd([3, 8, 5, 7, 1, 9, 2]))
```

```
[3, 5, 7, 1, 9]
```

Sample output of main function:

Original Array: [3, 8, 5, 7, 1, 9, 2]

Odd elements in the array: [3, 5, 7, 1, 9]

Java code

```
import java.util.ArrayList;
import java.util.List;
import java.util.Arrays;

public class spOdd {

public static List<Integer> SpOdd(int[] arr) {
    List<Integer> result = new ArrayList<>();
    for (int num : arr) {
        if (num % 2 != 0) {
            result.add(num);
        }
    }
    return result;
}

public static void main(String[] args) {
    int[] originalArray = {3, 9, 7, 4, 5, 1, 8, 5, 7, 1, 9, 2};

    System.out.println("Original Array: " + Arrays.toString(originalArray));

    List<Integer> oddElements = SpOdd(originalArray);
    System.out.println("Odd elements in the array: " + oddElements.toString());
}
}
```

Java code output

Original Array: [3, 9, 7, 4, 5, 1, 8, 5, 7, 1, 9, 2]

Odd elements in the array: [3, 9, 7, 5, 1, 5, 7, 1, 9]

QUESTION 4

Write Verilog code to describe the following functions:

f1=x1·(~x3)+x2·(~x3)+(~x3)·(~x4)+x1·x2+x1·(~x4) f2=(x1+(~x3))·(x1+x2+(~x4))·(x2+(~x3)+(~x4))

Verilog code

```
module boolean_functions(
input wire x1,
input wire x2,
input wire x3,
input wire x4,
output reg f1_output,
output reg f2_output
);

// Function f1
always @* begin
    f1_output = (x1 & ~x3) | (x2 & ~x3) | (~x3 & ~x4) | (x1 & x2) | (x1 & ~x4);
end

// Function f2
always @* begin
    f2_output = (x1 | ~x3) & (x1 | x2 | ~x4) & (x2 | ~x3 | ~x4);
end

endmodule
```

Test verilog code

```
module testbench;

// Inputs
reg x1, x2, x3, x4;
// Outputs
wire f1_output, f2_output;

boolean_functions dut (
    .x1(x1),
    .x2(x2),
    .x3(x3),
    .x4(x4),
    .f1_output(f1_output),
    .f2_output(f2_output)
);

// Stimulus generation
initial begin
    // test vectors
    x1 = 1; x2 = 0; x3 = 1; x4 = 0;
    #10;
    $display("x1=%b, x2=%b, x3=%b, x4=%b, f1_output=%b, f2_output=%b",
             x1, x2, x3, x4, f1_output, f2_output);
    $finish;
end

endmodule
```

Test output

x1=1, x2=0, x3=1, x4=0, f1_output=1, f2_output=1 main.v:48: $finish called at 10 (1s)

QUESTION 5

Load

```
Load the constant 23 into register 4
li $4, 23

Copy the contents of register 2 into register 3
move $3, $2

Load char (one byte) starting at memory address 244 into register 6
lb $6, 244($0)

Load R5 with the word whose memory address is in R1
lw $5, 0($1)

Load the word that begins 8 bytes after the address in R1. This is known as "constant offset"
mode and is about the fanciest addressing mode a RISC processor will support.
lw $2, 8($1)
```

Store the constant number 37 into the word beginning at 400

```
li $8, 37
sw $8, 400($0)
```

Store the value in R6 into the word whose address is in R1

```
sw $6, 0($1)
```

Store lower half-word from R2 into 2 bytes starting at address 1024

```
sh $2, 1024($0)
```

Store R7 into the word whose address is 12 more than the address in R1

```
sw $7, 12($1)
```

ALU

Add 6 to R3 and store the result in R1

```
addi $1, $3, 6
```

Subtract R3 from R2 and store the result in R1

```
sub $1, $2, $3
```

Branching

Begin executing at address 344 if R1 equals 0

```
beq $1, $0, 344($0)
```

Begin executing at addr 8 past current instruction if R2 less than R3

```
blt $2, $3, Label
<some instruction>
Label:
```