```
#Solutions for lab 4
#Suprakash Datta

#Q1:
printPair := proc(m, n)
 print("You entered ", m, " and ", n);
end proc;
```

```
proc(m, n) print("You entered ", m, " and ", n) end proc                    (1)
```

```
printPair(5, 6);
```

```
"You entered ", 5, " and ", 6                                               (2)
```

```
#Q2
myIrem := proc(m :: integer, n :: integer) # typechecking was not required for this assignment
 return m - n·trunc(m/n);
 end proc;
```

```
proc(m::integer, n::integer) return m - n*trunc(m/n) end proc               (3)
```

```
myIrem(2, 3);
```

```
2                                                                           (4)
```

```
myIrem(4, 1);
```

```
0                                                                           (5)
```

```
myIrem(37, 9);
```

```
1                                                                           (6)
```

```
#Q3:
largerNum := proc(n :: integer, m :: integer)
 return piecewise(m > n, m, m ≤ n, n);
 end proc;
```

```
proc(n::integer, m::integer) return piecewise(n < m, m, m <= n, n) end proc  (7)
```

```
largerNum(2, 2);
```

```
2                                                                           (8)
```

```
largerNum(4, 3);
```

```
4                                                                           (9)
```

```
largerNum(3, 4);
```

```
4                                                                           (10)
```

```
#Q4
TwoLargestNum :=proc(n :: integer, m :: integer, p :: integer)
 local largest, secondLargest, tmp;
 largest := largerNum(n, largerNum(m, p));
 if n = largest then secondLargest := largerNum(m, p);
 elif m = largest then secondLargest := largerNum(n, p);
 elif p = largest then secondLargest := largerNum(m, n);
 else print("Error"); secondLargest := -1;
 end if;
 print(largest, secondLargest);
```

```
 end proc;
proc(n::integer, m::integer, p::integer)                                    (11)
    local largest, secondLargest, tmp;
    largest := largerNum(n, largerNum(m, p));
    if n = largest then
        secondLargest := largerNum(m, p)
    elif m = largest then
        secondLargest := largerNum(n, p)
    elif p = largest then
        secondLargest := largerNum(m, n)
    else
        print("Error"); secondLargest := -1
    end if;
    print(largest, secondLargest)
end proc
```

```
TwoLargestNum(2, 39, 41);
                                    41, 39                                  (12)
```

```
#Q5
  q5proc :=proc(n)
 return piecewise(frac(n) = 0, print("The input is an integer"), frac(n) ≠ 0, evalf(n^2, 6));
 end proc;
proc(n)                                                                     (13)
    return piecewise(frac(n) = 0, print("The input is an integer"), frac(n) <> 0, evalf(n^2,
    6))
end proc
```

```
q5proc(4);
                            "The input is an integer"                       (14)
q5proc(4.32222);
                                   18.6816                                  (15)
#Note: In this case, ensuring printing to 5 decimal places is hard.You will get credit as long as you set
    the second field
 #of eval to some number bigger than or equal to 5

 #Q6
isInteger := proc(n)
 return piecewise(frac(n) = 0, 1, frac(n) ≠ 0, 0);
 end proc;
          proc(n) return piecewise(frac(n) = 0, 1, frac(n) <> 0, 0) end proc   (16)

isInteger(3);
                                       1                                    (17)
isInteger(-3);
```

$$1 \tag{18}$$

isInteger(*a*);

$$\begin{cases} 1 & \text{frac}(a) = 0 \\ 0 & \text{frac}(a) \neq 0 \end{cases} \tag{19}$$

isInteger(4.2);

$$0 \tag{20}$$

$$isInteger\left(\frac{4}{2}\right); \tag{21}$$

$$1$$

#Q7
with(*MTM*) : *log2*(8);

$$3 \tag{22}$$

powerOf2 := **proc**(*n* :: *integer*)
return isInteger(log2(n));
end proc;

$$\textbf{proc}(n::integer)\ \textbf{return}\ isInteger(MTM:\text{-}log2(n))\ \textbf{end proc} \tag{23}$$

powerOf2(4);

$$1 \tag{24}$$

powerOf2(5);

$$0 \tag{25}$$

?*if*