```
> #Solutions to practice problems
> #Q1
> myNextPrime :=proc(n :: posint)
    local num;
    num := n + 1;
    while true do #the termination condition is in the body of the loop
    if isprime(num) then return num;
    end if;
    num := num + 1; #need the  +1 because n may be a prime
    end do;
    end proc;
```

```
myNextPrime := proc(n::posint)                                    (1)
    local num;
    num := n + 1; do if isprime(num) then return num end if; num := num + 1 end do
end proc
```

```
> secprime :=proc(n :: posint)
    local num;
    num := myNextPrime(n);#first prime bigger than n
    return myNextPrime(num + 1);#second prime bigger than n
    end proc;
```

```
secprime := proc(n::posint)                                       (2)
    local num;
    num := myNextPrime(n); return myNextPrime(num + 1)
end proc
```

```
> secprime(5);
                                  11                              (3)
> secprime(11);
                                  17                              (4)
>
> #Q2
> #I changed the procedure to output the number of twin primes rather than the list to save space
> twinprime :=proc( )
    local num, count;
    count := 0;
    for num from 1 to 999998 do
    if isprime(num) = true and isprime(num + 2) = true then count := count + 1;
    end if;
    end do;
    print(count);
    end proc;
```

```
twinprime := proc( )                                              (5)
    local num, count;
    count := 0;
    for num to 999998 do
        if isprime(num) = true and isprime(num + 2) = true then count := count + 1 end if
    end do;
```

```
    print(count)
end proc
```

```
> twinprime( );
```

8169 **(6)**

```
>
> #Q3
> myfib :=proc(n :: posint)
  local num, L;
  L := [seq(0, i = 1 ..n) ];
  L[1] := 1;
  L[2] := 1;
  for num from 3 to n do
  L[num] := L[num − 1] + L[num − 2];
  end do;
  return L[n];
  end proc;
```

```
myfib := proc(n::posint)
    local num, L;
    L := [seq(0, i = 1 ..n) ];
    L[1] := 1;
    L[2] := 1;
    for num from 3 to n do L[num] := L[num − 1] + L[num − 2] end do;
    return L[n]
end proc
```

(7)

```
> myfib(5);
```

5 **(8)**

```
>
> #Q4
> nplusone :=proc(n :: posint)
  local num, curr;
  num := 0;
  curr := n;
  while curr > 1 do
  num := num + 1;
  #print(num, curr);
  if irem(curr, 2) = 0 then curr := curr/2;
  else curr := 3·curr + 1;
  end if;
  end do;
  return num;
  end proc;
```

```
nplusone := proc(n::posint)
    local num, curr;
    num := 0;
    curr := n;
    while 1 < curr do
```

(9)

```
        num := num + 1; if irem(curr, 2) = 0 then curr := 1/2 * curr else curr := 3 * curr + 1
        end if
    end do;
    return num
end proc
> nplusone(10000000000000);
                                    289                                    (10)
>
>
```