1.

```
Algorithm: pwFunc(x)
Requires: one real number x
Returns: the value of function f(x)
1:  if x>4
2:      return x+1
3:  elseif x>=0
4:      return x^0.5
5:  elseif x>-1
6:      return 3-2*x^2
7:  else
8:      return 0
9:  endif
```

2.

```
Algorithm: pwFunc(x)
Requires: one real number x
Returns: the value of function f(x)
1:  if x>=0
2:      if x>4
3:          return x+1
4:      else
5:          return x^0.5
6:      endif
7:  else
8:      if x>-1
9:          return 3-2*x^2
10:     else
11:         return 0
12:     endif
13:  endif
```

3.

```
Algorithm: isUnique(a, b, c)
Requires: three real numbers a, b, c
Returns: numbers indicate unique values in a, b, c
1:  if a==b
2:      if b==c
3:          return 3 // Case I
4:      else
5:          …… // Case II
6:      endif
7:  else
8:      if …….
9:          ……..// Case III
10:     else
```

```
11:        if a==c
12:           return 2 // Case IV
13:        else
14:           ……..// Case V
15:        endif
16:     endif
17:  endif
```

Note that this algorithm using nest IF has three layers i.e. indentation as required, identified by the following:

- Outer IF (Lines 1, 7, 17)
- Middle IF (Lines 2, 4, 6 and Lines 8, 10, 16)
- Inner IF (Lines 11, 13, 15)

Alternatively, use simple IF structure with compounded conditional statements:

```
1:  if a==b && b==c
2:    return 3
3:  elseif a==b || b==c || c==a
4:    return 2
5:  else
6:    return 1
7:  endif
```

4. In general, there are 5 different types of input arguments:
   I. a,b,c are equal
   II. a,b are equal, c is different
   III. b,c are equal, a is different
   IV. a,c are equal, b is different
   V. a,b,c are different

   These correspond to all the "return" actions taken in original nested IF structure. Hence, we can design five suitable input pairs accordingly. For example, we use isUnique(1,2,1) to test Case IV:

   Initially, a=1, b=2, c=1
   Line1: 1==2, False, go to Line 7
   Line 7: go inside the *else* block of outer IF
   Line 8: 2==1. False, go to line 10
   Line 10: go inside the *else* block of middle IF
   Line 11: 1==1. True, go to line 12
   Line 12: go inside the *if* block of inner IF, return 2.
   Line 15: close inner IF
   Line 16: close middle IF
   Line 17: close outer IF

   If you need to test algorithm like the alternative solution, you can roughly combine Cases II-IV and test them quickly with this well-structured algorithm.

5. use simple IF structure with compounded conditional statements:

6. i. sum = 9+(8*2-9)+3+(2*2) = 23, sum%10=3 Invalid
Similarly rest of the parts
……..

8. i. Trace alg1(365):

[Main]	*[Sub-algorithm]*
n = 365 1. Line 1: a = 5, d = 36 2. Line 2: b = 6, c = 3 3. Line 3: x = 14 4. Line 4: y = alg2(14) (call sub-algorithm)	n = 14 1. Line 1: 14>=10, True 2. Line 2: return 5 3. Line 5: endif.
= 5 (value returned from sub algorithm) 5. Line 5: 5==3 || 5==6 || 5==9 False, go to line 7 6. Line 7: else, go to line 8 7. Line 8: return False 8. Line 9: endif	

ii. This algorithm is to determine whether a three-digit integer n can be divided by 3.
For further information, you can read it here: Divisibility rule of 3.

9&10. Refer to discussions in Seminar 2 slides.