These are possible solution for the given questions you may have different method to accomplish task .

1 (a) Given numbers $X = 8$, $Y = 3$, and $Z = 5$.

Determine the values of the following expressions:

(i) X mod Y.
(ii) $X * Y / Z$.
(iii) $(X * Z)$ mod Y.

Solution 1 (i) 2 (ii) 4 (iii) 1

(b) Given a = true and b = false. State the result of:

```
if OR (AND(a,b),b) then
    return a
else
    return b
end if
```

(b) OR(AND(true ,false) ,false) = OR(false ,false) = false so else part executed and return b

i.e. false.

(c) A fashion store awards a discount to all of its customers over the Spring Holiday period, using the following policy.

If a customer's purchases reach 500 RMB or more they will get a discount of 10%, and for a purchase of 1000 RMB or more the customer will get a discount of 25% on the total bill. For smaller purchases a standard discount of 2% is given.

Write an algorithm which calculates the discount a customer will receive based on the total bill, and returns the amount to be paid after discount is subtracted.

1(c) **Algorithm: Sdiscount (bdb)**
Requires: Total Bill Amount of purchasing bdb>0.
Returns: Amount to pay after discount.

```
1. If(bdb >= 1000) then
2.     return (bbd *0.75)
3. else if (bdb >= 500) then
4.  return (bdb * 0.90)
5. else
6.     return (bbd * 0.98)
7. endif
```

* **Where bdb = bill before discount**

(d) Write an algorithm isEven that takes a number x as an argument and determines whether or not x is even. You may use addition, subtraction, equality tests, and recursion.
(You may NOT use any division or mod operators.)

(d) **Algorithm : isEven(X)**

Requires: a number X >=0

Returns: True if X is even; otherwise false

```
1. If(X == 0) then
2.  Return true
3. else
4. return NOT(isEven(X -1))
5. endif
```

Q2 (a)

Write a recursive algorithm **sumDownBy2** with the following specifications:

Requires: a positive integer n

Returns: a positive integer which is the sum $n + (n-2) + (n-4) + \ldots + 0$.

e.g. **sumDownBy2**(7) should return $7 + 5 + 3 + 1 = 16$.

sumDownBy2(8) should return $8 + 6 + 4 + 2 = 20$.

sumDownBy2(0) should return 0.

sumDownBy2(-1) should return 0.

Solution 2(a)

sumDownBy2(0)=0; sumDownBy2(-1)=0; sumDownBy2(7)= 7+ sumDownBy2(7-2)

```
Algorithm sumDownBy2(n)
Requires: a number, n
Returns: sum of n+(n-2)+..+1/2
    if    n==0       then
          return 0
    elseif n==-1  then
          return 0
    else
          return n+ sumDownBy2(n-2)
    endif
```

2(b) The least common multiple (LCM) of two numbers is the smallest multiple of the both. Write a recursive algorithm LCM with the following specifications. [3]

Requires: positive integers x and y
Returns: the least common multiple (LCM) of x and y
E.g. LCM $(3, 5)$ is 15
LCM $(6, 8)$ is 24.

```
Algorithm LCM(x,y)
Requires: 2 positive integers $x$ and $y$.
Returns: the least common multiple (LCM) of $x$ and $y$.
    return x*y/ gcd(x,y)

Algorithm gcd(x,y)
Requires: 2 positive integers $x$ and $y$.
Returns: The greatest common divisor of $x$ and $y$.
    if $x == y$ then
        return $x$
    else if $x > y$ then
        return gcd($x - y$, $y$)
    else if $x < y$ then
        return gcd($x$, $y - x$)
    endif
```

2(c) Write a recursive algorithm, **binL**, which takes a numbers n, as the argument. It should return another number with the same length, where all the digits with the even value have been changed to zero and the odd values to one.

E.g. **binL** (3416) should return **1010**.

Use $n = 3416$ to test the algorithm showing all of the intermediate steps. [4]

binL(5)=5 mod 2=1; binL(316)= (316/100 mod 2)*100+ binL (316 mod 100)

=(316/100 mod 2)*100+(16/10 mod 2)*10+0

```
Algorithm binL(n)
Requires: An integer n where most significant digit is odd number.
Returns: another number with the digits of 1 and 0
    if   n/10==0      then
        return n mod 2
    else
        let k=pvalue(n)
        return (n/k mod 2)* k +binL(n mod k)
    endif

Algorithm pvalue(n)
Requires: a number, n
Returns: eg. pvalue(316)=100, pvalue(32)=10
    if   n/10==0      then
        return 1
    else
        return 10* pvalue(n/10)
```

3 (a) A list L_1 is given as follow $[5, 9, 8]$.

I. Write this list explicitly using Nil and Cons.

Cons(5, Cons(9, Cons(8, Nil)))

II. Write pseudocode to obtain second element of list L_1

value(tail(L_1))

III. What is the result of isEmpty (tail (tail (tail (L_1))))?

True

3(b) Write an algorithm **countN** which takes a list of integers, ***list,*** and an integer N and returns the number of elements of list that are less than N.

E.g. **countN** $([2,3,3,7,6,8],7)$ returns 4

countN $([2,3,3,7,6,8],5)$ returns 3

countN $([2,3,3,7,6,8],2)$ returns 0

```
Algorithm countN (list,N)
Requires: a list and number N
Returns: the number of elements of list that are less than N
    if   isEmpty(list)          then
        return 0
    else
        if value(list)<N
            return 1+ countN (tail(list),N)
        else
            return countN (tail(list),N)
        endif
    endif
```

3(c) Write a recursive algorithm $\boldsymbol{split\_IV}$, that splits a list of numbers into a pair of two lists. The first list contains all the even values and the odd values are included in the second list.

E.g. given the list [9,4,3,5,2,7], the algorithm should return $([4,2],[9,3,5,7])$

```
Algorithm split_IV(list)
Requires: a list
Returns: a pair of two lists
    if   isEmpty(list)          then
        return (Nil, Nil)
    else
        let x=value(list)
        let (l1,l2)= split_IV(tail(list))
        if x mod 2==0
            return (Cons(x, l1), l2)
        else
            return (l1,Cons(x, l2))
        endif
    endif
```

3 (d) Write a recursive algorithm to determine whether or not any two numbers in a list can add to the value x.

E.g. list = $[3,5,6,7]$ and $x = 9$ the result is true, but for list = $[3,5,6,7]$ $x = 5$ or 10, it would be false.

Use the list $[3,6,7]$ and $x = 9$ to test the algorithm showing all of the intermediate steps.

```
Algorithm canAdd(list,x)
Requires: a list and number x
Returns: True if any two numbers in list can add to x or False otherwise
    if   isEmpty(list)          then
            return False
    elseif   isIncluded(tail(list),x-value(list))  then
            return True
    else
            return canAdd(tail(list),x)
    endif
```

```
Algorithm isIncluded(list,x)
Requires: a list and number x
Returns: True if x is in the list or False otherwise
    if   isEmpty(list)          then
            return False
    elseif   value(list)==x     then
            return True
    else
            return isIncluded(tail(list),x)
    endif
```

4(a) Write an algorithm **Zero** that takes a binary tree as the input and returns true if the tree contains the value 0, otherwise false. Your algorithm should work if the given tree is empty.

Algorithm Zero (t)

Requires: a binary tree, t

Returns: returns true if the tree contains the value 0, otherwise false.

```
if   isLeaf(list)   then
          return False
elseif \root(t) == 0      then
          return True
else
          return Zero(left(t)) OR  Zero(right(t))
endif
```

4 (b) Write an algorithm to check whether the given binary search tree is balanced or not. Test your algorithm for the binary search tree *t* (given below). Also show all intermediate steps.

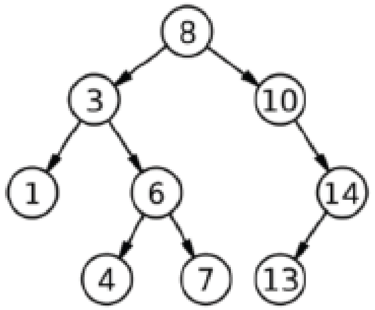


Figure -1: Binary search tree *t*

Algorithm: isBalanced(t)

Requires: a binary search tree t

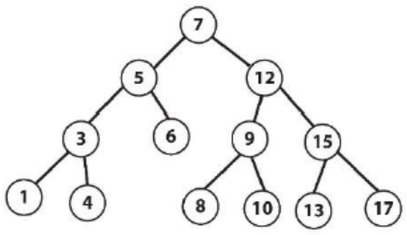
Returns: true if binary search tree is balanced otherwise false

```
1. if isleaf(t) then
2. return true
3. elseif max(depth(left(t)) ,depth(right(t))) -min(depth(left(t) ,
   depth(right(t)))<=1
4. return and (isBalanced(left(t)) , idsBalanced(right(t)))
5. else
6. return false
7. endif
```

```
Algorithm: depth(t)

Requires: a binary tree t

Returns: the depth of t

  1. if isleaf(t) then
  2. return 0
  3. else
  4. return 1 +max(depth(left(t)) ,depth(right(t)))
  5. endif
```

(c) The following operations are to be carried out in sequence on the binary search tree given below. [4]

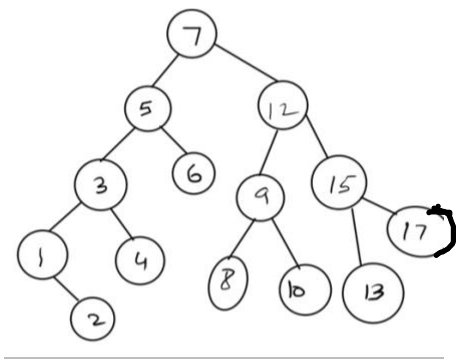
I. Add 2,
II. Add 11,
III. Delete 17,
IV. Delete 5.



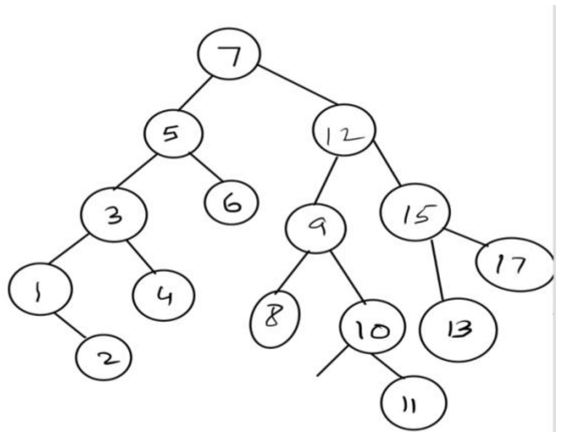
Draw a binary search tree structure after each operation in the sequence.

Note you should number each binary search tree I to IV and note that each new operation will be carried out on the previous tree.

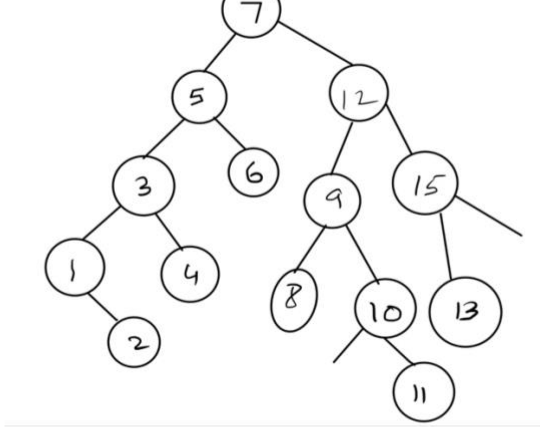
Solution 4 (c) [i]



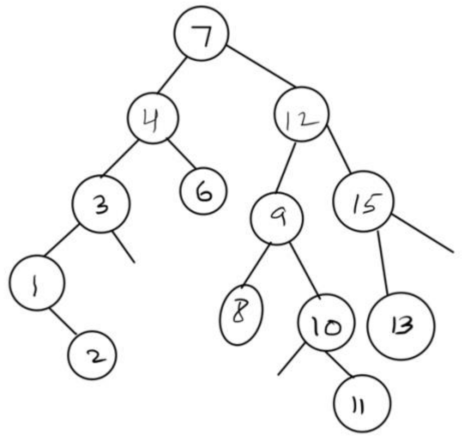
[ii]



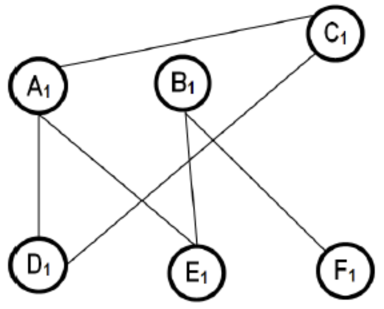
[iii]



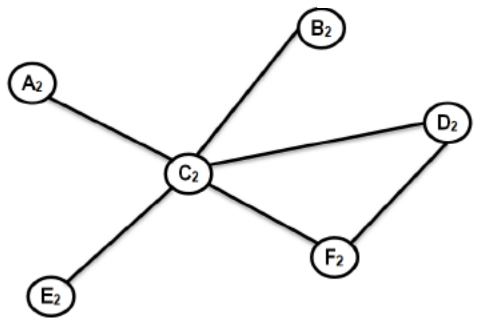
[iv]



5 (a) For each of the given graphs, determine whether the graph has a Euler path, a Euler tour, or neither. Justify your answer.



graph.1



graph.2

Euler path is possible for a graph1 but for graph2 neither Euler path nor Euler tour. Euler tour is only possible when degree of each vertex of graph is even. Euler path is possible if each vertex is of even degree except start and end vertex. Euler path is less restricted in comparison to Euler tour.

Note Euler tour /Euler circuit is same.

5 (b) Given directed graph:

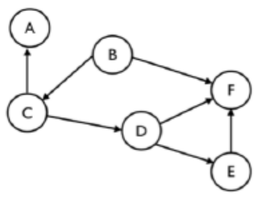


Figure-4

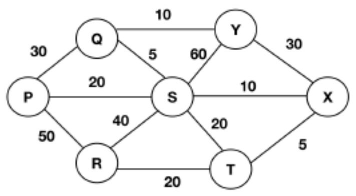
(i) Find any two possible topological sorting orders.

(ii) Which edge must be removed for the ordering ABCDEF to be a topological sorting order?

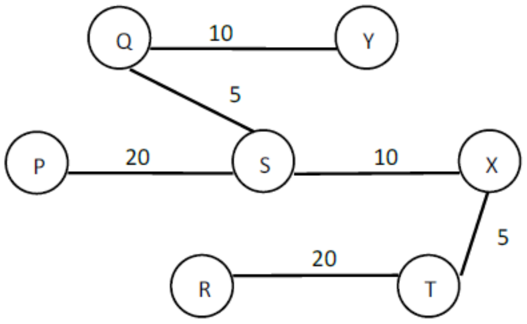
(i) **BCADEF and BCDEFA**

(ii) **Edge AC we need to remove to make topological sorting ABCDEF possible.**

5 (c) Find a minimum spanning tree for the given weighted graph.



Minimum spanning tree for given weighted graph is



Total weight of minimum spanning tree is 70