* **Question 3 (Algorithm)**

Have the function ArrayChallenge(arr) take the array of numbers stored in arr and return the string true if any combination of numbers in the array (excluding the largest number) can be added up to equal the largest number in the array, otherwise return the string false.

For example: if arr contains [4, 6, 23, 10, 1, 3] the output should return true because 4 + 6 + 10 + 3 = 23.

**Sol**

def ArrayChallenge(a):

    a = sorted(a,reverse = True)

    target = a[0]

    count = 0

    for x in a:

        if x == target:

            count += 1

    return subset(a, count - 1, len(a) - 1, target)

def subset(arr, i, n, s):

    if s == 0:

        return True

    if n == i:

        return False

    if arr[n] <= s:

        return subset(arr, i, n - 1, s - arr[n]) or subset(arr, i, n - 1, s)

    else:

        return subset(arr, i, n - 1, s)

# driver Program

print(ArrayChallenge (arr))

* **Question 2: (Database Designing)**

Design the database architecture for an inventory management solution to keep track of the current inventory.

The database design should support alerting when the inventory goes below the thresholds.

**Sol**

***Demand Table***

(PK)OrderID

CustomerID

ProductID

Quantity

OrderDate

***Customer Table***

(PK) CustomerID

CustomerMetaData

***Product Table***

**(PK)**Product ID (varchar)

Inventory Name (varchar)

StartingInventory (int)

InventoryReceived (int)

InventoryShipped (int)

InventoryInStore (int)

***Supplier Table***

(PK) SupplierID

SupplierName

Address

***Supply Table***

(PK) PurchaseID

SupplierID

ProductID

Quantity

PurchaseDate

* **Bonus Question What is the complexity of the following code:**

void function(int n)

{

int i = 1, s =1;

while (s <= n)

{

i++;

s += i;

n += 1;

n /= 1;

printf("\*");

}

}

**Answer –**

**Worst case complexity of the above code is O(n^0.5)**