Assignment 16

R 13.1 Professor Amongus has shown that a decision problem L is polynomial-time reducible to an NP-complete problem M. Moreover, after 80 pages of dense mathematics, he has also just proven that L can be solved in polynomial time. Has he just proven that P=NP? Why or why not?

R 13.3 Show that the problem SAT is NP-complete; SAT takes an arbitrary Boolean formula S as input and asks if S is satisfiable,.

R-13.13 Is there a subset of the numbers in {23, 59, 17, 47, 14, 40, 22, 8} that sums to 100? What about 130? Show your work.

A. Prove that the **Set-Partition** decision problem is a member of class **NP**. **Set-Partition** is defined as follows:

Set-Partition: Given a set **S** of integers, does there exist a partitioning of **S** into two disjoint partitions, such that the sum of the elements of both partitions is the same?

Hint: To be a partitioning, each element of S must be in either P1 or P2, but not both. Two partitions, P1 and P2 are disjoint if and only if no element S is a member of both P1 and P2. For example, suppose that S1 = {3, 6, 3}, then S1 can be partitioned into two partitions P1={3,3} and P2={6} whose sums are equal (6). However, S2={3, 5} cannot be partitioned in a way where the sums of two partitions are equal. Thus S1 is a member of the Set-Partition language, but S2 is not.

B. Recall the definition of Subset-Sum:

Subset-Sum: Given a set **S** of integers and a target integer **T**, does there exist a subset of **S** whose sum is equal **T**?

Below are four proposed reductions of **Subset-Sum** to **Set-Partition**; one is valid, but the other three are not. Determine which three proposed reductions are invalid and explain why with a counter example.

Hint: create two instances of Subset-Sum, one that has a subset that sums to T and another that does not. Then execute the three algorithms and it should be obvious which one is valid.

Explain why the other one is a valid reduction based on your instances of Subset-Sum.

```
Algorithm SS2SP_v1(S, T)
    sum ← 0
    for each i in S do
        sum ← sum + i
    S.insertLast(sum)
    return    S
```

```
Algorithm SS2SP_v2(S, T)
    sum ← 0
    for each i in S do
        sum ← sum + i
    S.insertLast(T)
    S.insertLast(sum-T)
    return  S

Algorithm SS2SP_v3(S, T)
    S.insertLast(T)
    return  S

Algorithm SS2SP_v4(S, T)
    sum ← 0
    for each i in S do
        sum ← sum + i
    S.insertLast(T)
    S.insertLast(sum+T)
    return  S

Algorithm SS2SP_v5(S, T)
    sum ← 0
    for each i in S do
        sum ← sum + i
    S.insertLast(sum – 2*T)
    return  S
```