Mo Felonda
Andrew Moy

CS577 HW10

Part A:

Input: A list of n courses S, their corresponding credits $c_1, \dots, c_n$ (where $c > 0 \mid c \in \mathbb{Z}$), the total sum of credits desired t, and the maximum number of courses allowed m.

Output: A list of the optimal classes that satisfies all requirements using the minimum number of classes B (B can be an empty set)

```
procedure MINIMUM-COURSES
base = MyUW(S,t,m)
if base == false then
    return B
k=m
S'=S
for i=1...n do
    S' = S'\{i}
if MyUW(S',t,k) == false, then
    Add course i to A
    t=t-c_i
    k = k-1
end if
    S'=S
end for loop
return B
```

Proof: Suppose we have a set of q courses $a_1, \dots, a_q$ that add up some d credits, where q is the minimum number of courses required to do so. Suppose we pass this set to the tool MyUW; we know that it would return true, because the set has $\leq q$ courses that add up to d credits. Suppose we have a set of p courses $a_1, \dots a_p$ that do not satisfy the requirements of the prompt (i.e. the number of courses exceeds the maximum allowed, or the sum of credits is not equal to that which is desired). Suppose we pass this set to MyUW; we know that it would return false, because the number of courses exceeds q or the sum of credits are unequal to d.

Time Complexity: It takes O(n) time to iterate through S. Calls to MyUW, decrements, and assignments are all done in constant time. Therefore the overall time complexity of this algorithm is O(n), which is polynomial in n.

Part B

Input: A list of the amount of credits that each class is worth $a_1, \dots, a_g$, the number of desired credits T, and the maximum number of courses that can be taken to achieve T, m.

Output: A boolean variable bestSchedule that returns true when T credits are reached when taking at most m courses.

Reduction: Multiply all elements in the list by F, where F is equal to 10^n, where n is equal to the least z such that 10^z is greater than the number of elements in the list. Then increment all elements in the list by 1. Then add $g-1$ dummy variables to the list, such that each dummy variable is equal to F. Let v be equal to $(T*F)+m$. Pass the list and v to the subset sum problem. If the subset sum problem returns true, set bestSchedule equal to true. If it returns false, set bestSchedule equal to false. Finally, return bestSchedule.

Correctness: Given some $I \subseteq [g]$ that satisfies $\sum_{i \in I} a_i = T$ where $|I| \leq m$, then $\sum_{i \in I} c_i = \sum_{i \in I} (Fa_i + m)$. Thus, $\sum_{i \in I} c_i = (F\sum_{i \in I} a_i) + m = (T*F) + m = v$. Assume that if some $J \subseteq [g]$, then $\sum_{j \in J} c_j = v$. More formally, $F\sum_{j \in J} a_j + m = (T*F) + m$.
F is defined as it is so that it is larger than g (the number of elements in the array), therefore it must be true that $m \leq g < F$. All variables here are representative of nonnegative integers, so dividing both sides of the formal equation by F and subtracting m reveals that $\sum_{j \in J} a_j = T$. Therefore, $I = J$ is a valid solution to this problem. Note that the inclusion of dummy variables is for the purpose of cases in which the required number of courses to reach T credits is less than the maximum number of courses that can be taken.

Time Complexity: Iterating through elements so that they can be multiplied takes $O(g)$. The multiplication itself, all comparisons, incrementing, addition of dummy variables, assignments, and calls to the subset sum problem can be done in constant time. Thus the overall time complexity is $O(g)$. Note that this time complexity is equivalent to $O(\#\text{ of courses})$.

(inspiration taken from discussion problem 2)