CSE 332: Data Abstractions

Section 1: `WorkList`s Solutions

`WorkList` ADT

`add(`**work**`)`	Notifies the worklist that it must handle **work**
`peek()`	Returns the next item to work on
`next()`	Removes and returns the next item to work on
`hasWork()`	Returns true if there's any work left and false otherwise

0. Interview Question: The Missing Number

Suppose `nums` is a `WorkList` of size n which contains each number between 1 and n exactly once. A user calls `nums.next()` but forgets to save the value. Recover the value that was removed. Can you do it if two values are removed? What are the time and space complexity of your solution?

Solution:

```
1  findRemoved(nums) {
2    sum = n * (n + 1) / 2;
3    while (nums.hasWork())
4       sum = sum - nums.next();
5    return sum;
6  }
```

To see why this works, note that the sum of all the values in `nums` before the user calls `nums.next()` is $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$. After the call to `nums.next()`, the corresponding sum is $\sum_{i=1}^{n} i - a = \frac{n(n+1)}{2} - a$, where a is the first number in `nums`. Therefore, the difference of $\frac{n(n+1)}{2}$ and the latter sum is exactly a, which is what the `findRemoved` function returns.

To find two removed values, we need to make two "measurements". One way to do so is as follows.

```
1  findTwoRemoved(nums) {
2    s1 = n * (n + 1) / 2;
3    s2 = n * (n + 1) * (2n + 1) / 6;
4    while (nums.hasWork()) {
5       t = nums.next();
6       s1 = s1 - t;
7       s2 = s2 - t * t;
8    }
9    return (s1 ± √(2s2 - s1^2))/2;
10 }
```

Note that if the two removed values are a and b, then at the end of the while loop (in line 8), we have $\texttt{s1} = a + b$ and $\texttt{s2} = a^2 + b^2$. Solving for a, b we get $\frac{\texttt{s1} \pm \sqrt{2\texttt{s2} - \texttt{s1}^2}}{2}$.

In both solutions we keep $O(1)$ numbers and perform $O(1)$ multiplications and additions per element.

1. Subset Sum

The `subsetSum` problem is:

> Given a number `sum` and an `int[] arr`, is there a subset of the elements in `arr` that sum to `sum`?

Fill in this recursive solution:

```
1  subsetSum(sum, idx, arr) {
2     if (sum == 0) {
3        return [ true ];
4     }
5
6     if (idx == arr.length) {
7        return [ false ];
8     }
9
10    return subsetSum( [ sum ], [ idx + 1 ], arr) ||
11           subsetSum( [ sum - arr[idx] ], [ idx + 1 ], arr);
12 }
```