1 (6 points) Write a program to implement Selection($T[1..n]$, $s$). Describe your computer.
   a Determine the expected execution time of Selection($T[1..n]$, $n/2$) on your computer.
   b Determine the expected execution time of Selection($T[1..n]$, 1) on your computer.

2. (8 points) Write a program to implement Quicksort($T[i..j]$).
   a Determine the time to Quicksort a random permutation of $n$ elements on your computer.
   Change pivot (or partition from class) so that whenever $j - i + 1 > k$, instead of pivoting around $T[i]$ it pivots around the median of $k$ randomly selected elements of $T[i..j]$.
   b Determine the value of $k$ for your implementation for which Quicksort($T[i..j]$) is as fast as possible.
   c Determine the time to Quicksort a random permutation of $n$ elements on your computer such that you use the modified pivot with the optimal value of $k$.

3. (6 points) A typical dynamic programming algorithm provides the cost of a solution or establishes the existence of a solution without actually constructing the solution. To see how to construct a solution by using an efficient mechanism which tests for the existence of a solution, solve the following:
   You may call a boolean function BlackBox of two inputs:
   -a list of integers $x_1, x_2, ..., x_n$,
   -an integer $q$.
   and you are told that, in time in $O(1)$, BlackBox will return true if there is some subset of $x_1, x_2, ..., x_n$ whose sum is $q$ and false otherwise. Design an algorithm (a program is not needed) with the same input which will return an actual subset of $x_1, x_2, ..., x_n$ whose sum is $q$, if such a subset exists, or it should return failure.
   For example, BlackBox( (23,27,41,72,-4,6), 29) would return true, but your algorithm with the same input would return (27,-4,6) or (23,6). Your algorithm may call BlackBox as often as it wishes and it should work in time in $O(n)$. 