COMP3121
Assignment 4
A17S1N3

Evangelos Kohilas
z5114986

By submitting this document you are confirming that all the answers are your work and are not take from any other sources unless clearly mentioned.

Question 1

To create the longest subsequence of elephants, we first sort the elephants in decreasing order of IQs[1][2].

Then, we find the longest increasing subsequence of weight from our collection of sorted elephants.

Let S be our sorted collection, with $S[i]$ denoting the i^{th} elephant in the sequence.
Let $W(i)$ be the given weight of elephant i.

Then, for each elephant i in the collection, we let our subproblem be finding a subsequence of maximum length such that $W(i)$ is increasing and the subsequence ends with elephant i.

We do this process recursively, assuming we have solved all the subproblems for some $j < i$, and we look for all elephants m such that $W(m) \leq W(i)$ and $S[m] \leq S[i]$, picking elephant m which produced the longest increasing subsequence ending with m, and extending it to our subproblem to obtain the longest increasing subsequence ending with elephant i

Our solution is optimal as truncating the solution i will produce the optimal solution of the subproblem m.

[1] Sorting by the weight will also achieve the same result provided the method is inversed.
[2] If two elephants have the same value, we then sort those elephants by the respective inverse of the other value.

Question 2

Let A be the sequence of wooden peices that will be cut from dividing the wooden stick respective to its marks, with $A[i]$ denoting the size of the i^{th} peice in the sequence.
Let n be the total number of respective peices.

Let the subproblems to be considered as $c(i, j)$ be the minimum of the group of two or more peices, and the group of the remaining peices. This group signifies a group to that will be "joined back together", to simulate the cutting cost.

The recursion:

$$c(i,j) = \min \begin{cases} A[i] + A[j], & \text{if } j - i = 1 \\ c(i,k) + c(k+1,j) & \text{otherwise} \end{cases} : i < k < j - 1$$

Using this, we then examine every possible way of placing the outermost groupings, which follows from above when $c(0, n)$. Doing so will calculate the solutions of all the subproblems, however some will have already been calculated and stored in slots (i, j) of our table.

Since we are examining every possible grouping, our solution is trivially optimal.

Question 3

Let A be a sequence of Roman letters, with $A[i]$ denoting the i^{th} letter of A.
Let B be a sequence of Roman letters, with $B[i]$ denoting the i^{th} letter of B.

Assuming our base cases are solved, we fill our table T, for all $0 \leq a < |A|$ and all $0 \leq b < |B|$ (row by row) as shown in the recursion:

$$s(a,b) = \begin{cases} s(a,b+1), & \text{if } A[a] \neq B[b] \\ s(a,b+1) + s(a+1,b+1) & \text{if } A[a] = B[b] \end{cases}$$

Finally, the number of sequences found will then be stored in the bottom right of the table, $T[|A|-1][|B|-1]$.

Since we are iterating over every combination of characters, our solution is trivially optimal.

Question 4

Let each supervisor be a node. Let the supervisors' employees' be the node's children. Let the fun rating of an employee be value.

We then let our subproblem be the maximum of either the total maximum of a node's children, or total of the node's value and the maximum of each children's children.

We do this process recursively, assuming we have solved all the subproblems for some i. By picking the maximal set of positioned nodes, we are always extending our original subproblem with the largest total value to obtain an increasing total value that must end at the current node.

Thus, our solution is optimal as truncating the solution i will still produce the optimal solution of the subproblem m.

Question 5

$$r(x,y,z) = \begin{cases} True & \text{if we've reached the end of all the strings} \\ r(x+1,y,z+1), & \text{if } X[x] = Z[z] \\ r(x,y+1,z+1), & \text{if } Y[y] = Z[z] \\ False & \text{otherwise} \end{cases}$$

Question 6

Let our collection of turtles be denoted by T, with T_i denoting the i^{th} turtle.
Let the weight and strength of the turtle T_i be $W(T_i)$ and $S(T_i)$ respectively.

Then for each turtle in T, we let our subproblem be finding the longest chain of turtles such that each turtle's strength is larger than or equal to the total weight of all turtles above it.

However a longest chain may not have the last turtle included as the lowest turtle in the tower, thus our chains cannot be extensions of one another.

Instead we define a new subproblem to find the lightest turtle tower of length l such that no turtles will be cracked, and we define our ordering to use the sum of a turtle's weight and strength, denoted by $W(T_i)+S(T_i)$

We claim that if there exists a tower of length l, then there must exist a tower of the same length such that $W(T_i) + S(T_i) \leq W(T_{i+1}) + S(T_{i+1})$ for all $i < l$. By proving this, and showing that the swapping of two adjacent unordered turtles where $W(T_{i+1}) + S(T_{i+1}) \leq W(T_i) + S(T_i)$ such that they will preserve our constraints, we can then BubbleSort our stack approrpiately.

Assume turtles T_i and T_{i+1} are inverted in our stack such that T_{i+1} is lower in the stack, and $W(T_i) + S(T_i) > W(T_{i+1}) + S(T_{i+1})$, then:

$$\sum_{j=1}^{i} W(T_j) \leq S(T_{i+1})$$

Now we add turtle T_{i+1}

$$\sum_{j=1}^{i+1} W(T_j) \leq S(T_{i+1}) + W(T_{i+1}) < S(T_i) + W(T_i)$$

Then expand the sum

$$\sum_{j=1}^{i-1} W(T_j) + W(T_i) + W(T_{i+1}) < S(T_i) + W(T_i)$$

Cancelling our the common terms, we then conclude that

$$\sum_{j=1}^{i-1} W(T_j) + W(T_{i+1}) < S(T_i)$$

such that the left hand side will be what the turtle T_i will see, which is less than the $S(T_i)$. Therefore the weight of the turtles above the swapped turtle's is less than it's strength, and thus our turtles will not crack if they are swapped.

Thus, by using the above and building the lightest possible tower of every possible length, we can then obtain a solution for any tower t by picking the longest tower obtained by solving t.