Problem 1

Result in COE Linux:

```
-bash-4.2$ ./1.out
Value  44  find in the vector at position: 7
The first element larger than 50 at position: 1
Not all elements in vector are greater than 45.
Print vector:
54 58 46 56 48 60 44 50 52 42
Print list:
44 48 36 46 38 50 34 40 42 32
Print stack:
22 32 30 24 40 28 36 26 38 34
Print priority_queue:
30 28 26 24 22 20 18 16 14 12
```

Problem 2

Result in COE Linux:

```
-bash-4.2$ ./2.out
Rod size: 5
Use recursion: 12
It took 0 clicks (0.000000 seconds).
Use DP: 12
It took 0 clicks (0.000000 seconds).

Rod size: 10
Use recursion: 25
It took 0 clicks (0.000000 seconds).
Use DP: 25
It took 0 clicks (0.000000 seconds).

Rod size: 15
Use recursion: 37
It took 0 clicks (0.000000 seconds).
Use DP: 37
It took 0 clicks (0.000000 seconds).

Rod size: 20
Use recursion: 50
It took 10000 clicks (0.010000 seconds).
Use DP: 50
It took 0 clicks (0.000000 seconds).

Rod size: 25
Use recursion: 62
It took 560000 clicks (0.560000 seconds).
Use DP: 62
It took 0 clicks (0.000000 seconds).

Rod size: 30
Use recursion: 75
It took 17360000 clicks (17.360001 seconds).
Use DP: 75
It took 0 clicks (0.000000 seconds).

Rod size: 35
Use DP: 87
It took 0 clicks (0.000000 seconds).

Rod size: 40
Use DP: 100
It took 0 clicks (0.000000 seconds).

Rod size: 45
Use DP: 112
It took 0 clicks (0.000000 seconds).

Rod size: 50
Use DP: 125
It took 0 clicks (0.000000 seconds).
```

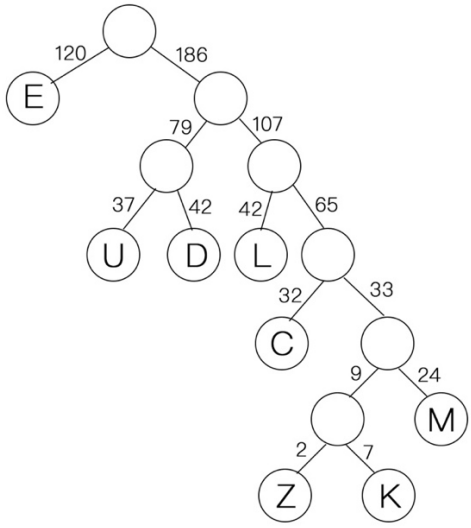
When rod size is 35, the running time is more than 500s, so recursive is not used in longer size.

Because COE Linux is so fast, the processing time is almost 0 when using DP in situation of 50. The following data is from Xcode IDE on my MacBook, which is relatively slower than COE Linux, but it still shows that DP is faster than Recursive.

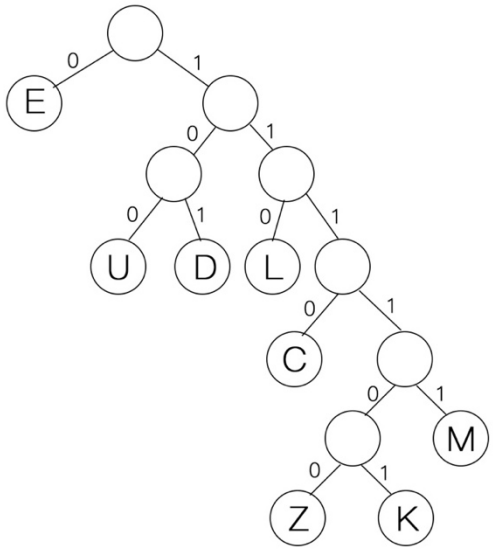
Rod Size	**Recursive Time (s)**	**Recursive Max Revenue**	**Dynamic Time (s)**	**Dynamic Max Revenue**
5	0.000003	12	0.000002	12
10	0.000023	25	0.000003	25
15	0.000719	37	0.000006	37
20	0.016127	50	0.000004	50
25	0.509955	62	0.000009	62
30	16.024006	75	0.000009	75
35	>3 mins	/	0.000009	87
40	>3 mins	/	0.000025	100
45	>3 mins	/	0.000021	112
50	>3 mins	/	0.000019	125

Problem 3

The following picture shows the weight display of the process of constructing a Huffman tree.



This picture shows the Huffman tree with the left branch of the weight changed to 0 and the right branch changed to 1.



For the eight letters to be coded with 0 or 1 of the path from the root to the leaf, the following table can be obtained.

Letter	E	U	D	L	C	M	Z	K
Huffman code	0	100	101	110	1110	11111	111100	111101