Question 1:

a)

	Start10	Start12	Start20	Start30	Start40
ucsdijkstra	2565	Mem	Mem	Mem	Mem
ideepsearch	2407	13812	5297410	Time	Time
astar	33	26	915	Mem	Mem
idastar	29	21	952	17297	112571

b)

1. According to this table above, we can know that **IDA* Search** is the most efficient algorithm in this four, since it does not appear any out of global stack error and runtime error from start10 to start40.
2. Compared with other three searching algorithm, **A* Search** is the second efficient because of the fewer node, although it has memory error in start30 and start40.
3. **Iterative Deepening Search** is not efficient since it generates a lot of nodes and runtime error in Start30 and Start40, but it is better than **Uniformed Cost Search**.
4. From this table, **Uniformed Cost Search** is not efficient algorithm, because it not only generates overmuch nodes in start10, and it causes memory error in the rest of the test.

Question 2:

a)

The answer is from the table below.

	Start50		Start60		Start64	
IDA*	50	14642512	60	321252368	64	1209086782
1.2	52	191438	62	230861	66	431033
1.4	66	116342	82	4432	94	190278
1.6	100	33504	148	55626	162	235848
Greedy	164	5447	166	1617	184	2174

b)

```
depthlim(Path, Node, G, F_limit, Sol, G2)  :-
    nb_getval(counter, N),
    N1 is N + 1,
    nb_setval(counter, N1),
    % write(Node),nl,   % print nodes as they are expanded
    s(Node, Node1, C),
    not(member(Node1, Path)),      % Prevent a cycle
    G1 is G + C,
    h(Node1, H1),
    F1 is 0.8*G1 + 1.2*H1,
    F1 =< F_limit,
    depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

c)
The answer is from the table above.

$w = 1.4$: `F1 is 0.6*G1 + 1.4*H1,`

$w = 1.6$: `F1 is 0.4*G1 + 1.6*H1,`

d)
Greedy Search and **IDA* Search** are both stable algorithms, since it does not cause any memory or runtime error from start50 to start 64.
According to the table above and formula $f(n) = (2 - w) * g(n) + w * h(n)$, the number of nodes is the larger than other four situation when $w = 1$, but the length of path is the least.
From $w = 1.2$, $w = 1.4$ and $w = 1.6$, we can get that if w closes to 2, the number of nodes is reducing, and the length of path is increasing.
Though **Greedy Search** is not complete and optimal, it uses less time than **IDA* Search**. As a result, if we need to get the least path, w should be 1. However, although we give up the optimal solution, the speed of run can be faster when w is increasing.

Question 3:

a)

$$h(x, y, x_G, y_G) = |x - x_G| + |y - y_G|$$

b)
(i)

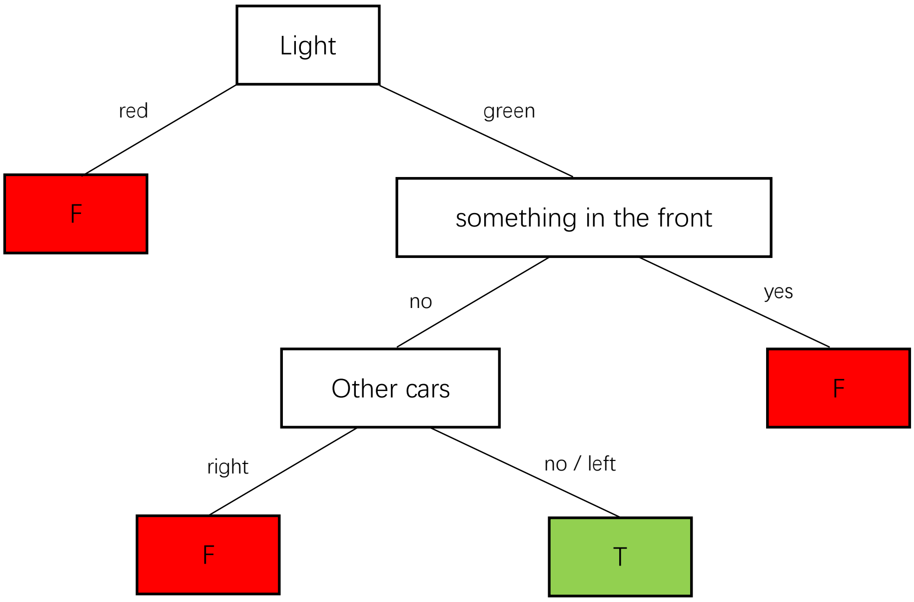
No. If we need to move the point from (1,1) to (5,6), we can get $h_{SLD}(x, y, x_G, y_G) = \sqrt{(x - x_G)^2 + (y - y_G)^2} = \sqrt{(1-5)^2 + (1-6)^2} = \sqrt{41}$. Actually, the true cost ($h^*(n)$) from (1,1) to (5,6) is 5. Thus, because of $h_{SLD}(n) > h^*(n)$, the Straight-Line-Distinct heuristic is not admissible.

(ii)
No. Using the same example as the part (b) first question, we also get $h(x, y, x_G, y_G) = |x - x_G| + |y - y_G| = |1-5| + |1-6| = 9 > h^*(n)$. So, the heuristic from part (a) is also not admissible.

(iii)

$$h(x, y, x_G, y_G) = \max(|x - x_G|, |y - y_G|)$$

Question 4:



1. Considering that traffic light is red or green, agent cannot pass the road if the light is red. If traffic light is green is green, agent needs think about another situation.
2. If there are something such as pedestrians and cyclists in the front of agent, agent cannot move forward.
3. If there are other cars in the right road, agent has to give way to those cars.
4. Agent can move forward, if there are no other cars or other cars in left road.

In my opinion, this case is unable to use in an agent since there are a lot of unexpected situation and random forecast. For example, if there is an ambulance or fire truck, the agent has to give a way for them.

Question 5:

a)

These tables below are to determine whether customers have ability to repay the loans.

i) two attributes:

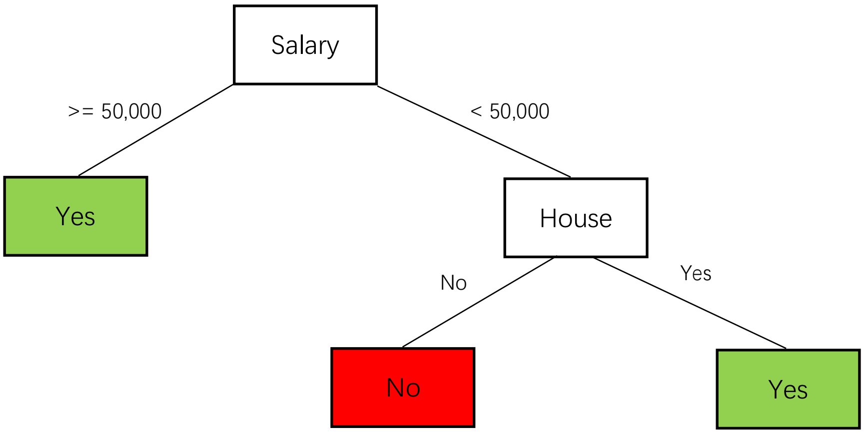
ID	Salary	House	Class
1	60,000	Yes	Yes
2	40,000	No	No
3	100,000	No	Yes
4	40,000	Yes	Yes

$$Entropy_{(S)} = -\frac{3}{4} \times log_2\frac{3}{4} - \frac{3}{4} \times log_2\frac{3}{4} \approx 0.81128$$

$$Gain_{(Salary,\ S)} = Entropy_{(S)} - (\frac{1}{2} \times 0 + \frac{1}{2} \times 1) \approx 0.31128$$

$$Gain_{(House,\ S)} = Entropy_{(S)} - (\frac{1}{2} \times 0 + \frac{1}{2} \times 1) \approx 0.31128$$

Since $Gain_{(Salary,\ S)} = Gain_{(House,\ S)}$, either salary or house can be as the first node.



ii) three attributes:

ID	Marital Status	Salary	House	Class
1	Single	60,000	Yes	Yes
2	Single	40,000	No	No
3	Married	100,000	No	Yes
4	Married	40,000	Yes	Yes
5	Single	100,000	No	Yes
6	Married	30,000	No	Yes

$$Entropy_{(S)} = -\frac{5}{6} \times log_2\frac{5}{6} - \frac{1}{6} \times log_2\frac{1}{6} \approx 0.65002$$

$$Entropy_{(Single)} = -\frac{2}{3} \times log_2\frac{2}{3} - \frac{1}{3} \times log_2\frac{1}{3} \approx 0.918296$$

$$Entropy_{(Married)} = 0$$

$$Gain_{(MS,\ S)} = Entropy_{(S)} - (\frac{1}{2} \times Entropy_{(Single)} + \frac{1}{2} \times 0) \approx 0.19087$$

$$Entropy_{(Salary \geq 50{,}000)} = 0$$

$$Entropy_{(Salary<50{,}000)} = -\frac{2}{3} \times log_2\frac{2}{3} - \frac{1}{3} \times log_2\frac{1}{3} \approx 0.918296$$

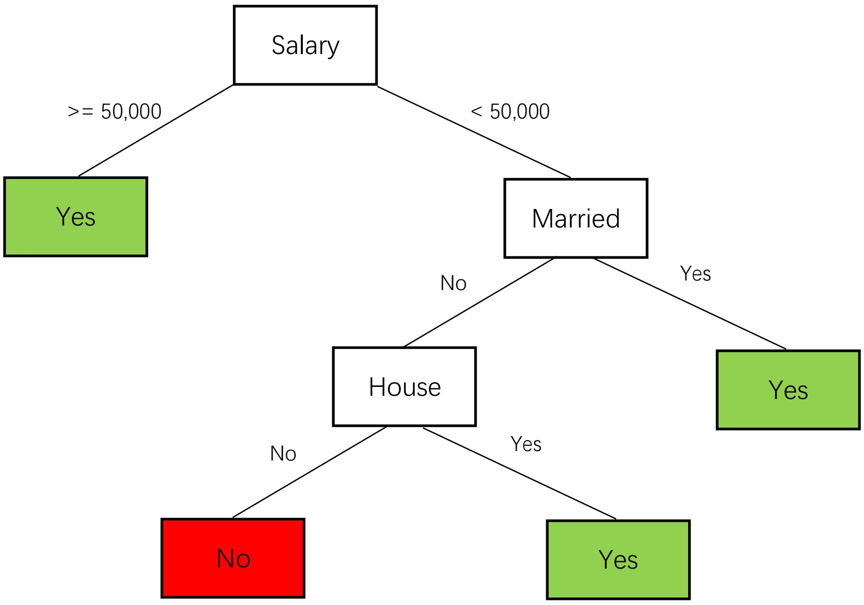
$$Gain_{(Salary,\ S)} = Entropy_{(S)} - (\frac{1}{2} \times Entropy_{(Salary<50{,}000)} + \frac{1}{2} \times 0) \approx 0.19087$$

$$Entropy_{(House=yes)} = 0$$

$$Entropy_{(House=no)} = -\frac{3}{4} \times log_2\frac{3}{4} - \frac{1}{4} \times log_2\frac{1}{4} \approx 0.811278$$

$$Gain_{(House,\ S)} = Entropy_{(S)} - (\frac{2}{6} \times 0 + \frac{4}{6} \times Entropy_{(House=no)}) \approx 0.109170$$

Since $Gain_{(MS,\ S)} = Gain_{(Salary,\ S)} > Gain_{(House,\ S)}$, we can get decision tree below:



iii) four attributes:

ID	Marital Status	Salary	House	Credit	Class
1	Single	60,000	Yes	Good	Yes
2	Single	40,000	No	Bad	No
3	Married	100,000	No	Bad	No
4	Married	40,000	Yes	General	Yes
5	Single	100,000	No	General	Yes
6	Married	30,000	No	Good	Yes
7	Single	55,000	Yes	General	Yes
8	Single	45,000	No	General	No
9	Single	40,000	Yes	Good	Yes
10	Married	38,000	Yes	General	Yes
11	Married	41,000	No	Good	Yes
12	Single	33,000	Yes	Bad	No
13	Single	52,000	No	Good	Yes
14	Married	31,000	Yes	General	Yes
15	Single	70,000	No	Bad	No
16	Married	81,000	Yes	Bad	No

$$Entropy_{(S)} = -\frac{10}{16} \times log_2 \frac{10}{16} - \frac{6}{16} \times log_2 \frac{6}{16} \approx 0.954434$$

$$Entropy_{(Single)} = -\frac{5}{9} \times log_2 \frac{5}{9} - \frac{4}{9} \times log_2 \frac{4}{9} \approx 0.991076$$

$$Entropy_{(Married)} = -\frac{5}{7} \times log_2 \frac{5}{7} - \frac{2}{7} \times log_2 \frac{2}{7} \approx 0.863120$$

$$Gain_{(MS,\ S)} = Entropy_{(S)} - (\frac{9}{16} \times Entropy_{(Single)} + \frac{7}{16} \times Entropy_{(Married)}) \approx 0.019338$$

$$Entropy_{(Salary \geq 50{,}000)} = -\frac{4}{6} \times log_2 \frac{4}{6} - \frac{2}{6} \times log_2 \frac{2}{6} \approx 0.918296$$

$$Entropy_{(Salary < 50{,}000)} = -\frac{6}{10} \times log_2 \frac{6}{10} - \frac{4}{10} \times log_2 \frac{4}{10} \approx 0.970951$$

$$Gain_{(MS,\ S)} = Entropy_{(S)} - (\frac{6}{16} \times Entropy_{(Salary \geq 50{,}000)} + \frac{10}{16} \times Entropy_{(Salary < 50{,}000)}) \approx 0.003229$$

$$Entropy_{(House=yes)} = -\frac{6}{8} \times log_2 \frac{6}{8} - \frac{2}{8} \times log_2 \frac{2}{8} \approx 0.811278$$

$$Entropy_{(House=no)} = 1$$

$$Gain_{(House,\ S)} = Entropy_{(S)} - (\frac{8}{16} \times Entropy_{(House=yes)} + \frac{8}{16} \times Entropy_{(House=no)}) \approx 0.048794$$

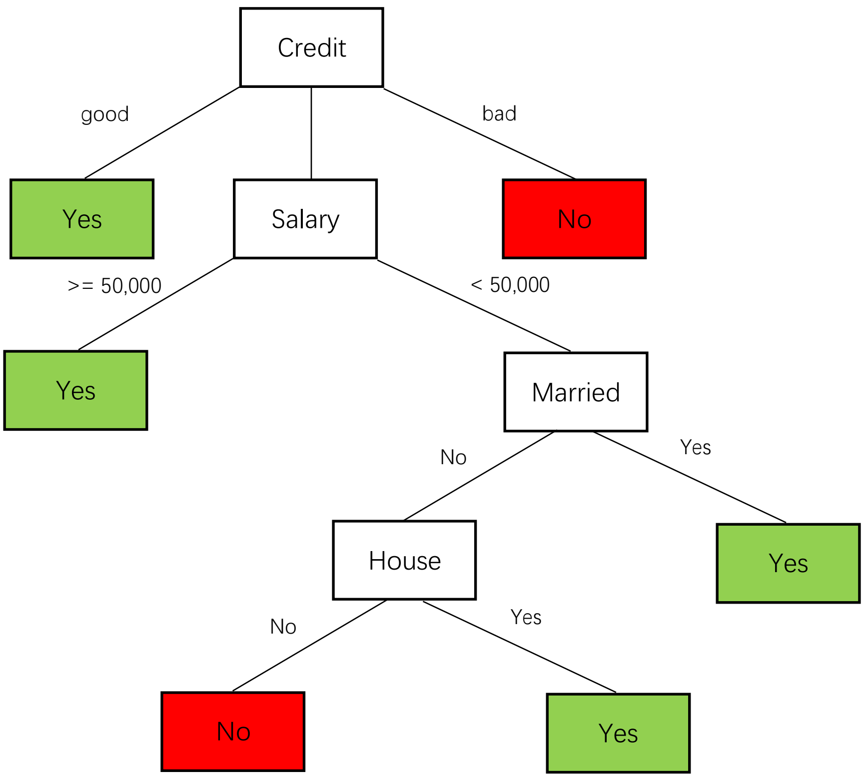
$$Entropy_{(Credit=good)} = 0$$

$$Entropy_{(Credit=\text{general})} = -\frac{5}{6} \times log_2 \frac{5}{6} - \frac{1}{6} \times log_2 \frac{1}{6} \approx 0.650022$$

$$Entropy_{(Credit=bad)} = 0$$

$$Gain_{(Credit,\ S)} = Entropy_{(S)} - (\frac{5}{16} \times Entropy_{(Credit=good)} + \frac{6}{16} \times Entropy_{(Credit=general)} + \frac{5}{16} \times Entropy_{(Credit=bad)}) \approx 0.710675$$

Thus, 'Credit' gives us the maximum information gain. We can get decision tree below:



b)

In my opinion, this learning algorithm could not return absolutely correct tree. However, the algorithm can generate tree which is logically equivalent since there are many different trees can be generated by the same training-set example and different methods.