2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

**CANDIDATE-ELIMINATION Learning Algorithm**

The CANDIDATE-ELIMINATION algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.

- Initialize G to the set of maximally general hypotheses in H
- Initialize S to the set of maximally specific hypotheses in H
- For each training example d, do
  - If d is a positive example
    - Remove from G any hypothesis inconsistent with d
    - For each hypothesis s in S that is not consistent with d
      - Remove s from S
      - Add to S all minimal generalizations h of s such that
        - h is consistent with d, and some member of G is more general than h
        - Remove from S any hypothesis that is more general than another hypothesis in S
  - If d is a negative example
    - Remove from S any hypothesis inconsistent with d
    - For each hypothesis g in G that is not consistent with d
      - Remove g from G
      - Add to G all minimal specializations h of g such that
        - h is consistent with d, and some member of S is more specific than h
        - Remove from G any hypothesis that is less general than another hypothesis in G

**CANDIDATE- ELIMINATION algorithm using version spaces**

**Training Examples:**

<table>
<thead>
<tr>
<th>Example</th>
<th>Sky</th>
<th>AirTemp</th>
<th>Humidity</th>
<th>Wind</th>
<th>Water</th>
<th>Forecast</th>
<th>EnjoySport</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Rainy</td>
<td>Cold</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Change</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Cool</td>
<td>Change</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Program:

```python
import csv

def g_0(n):
    return ('?',)*n

def s_0(n):
    return ('0',)*n

def more_general(h1, h2):
    more_general_parts = []
    for x, y in zip(h1, h2):
        mg = x == '?' or (x != '0' and (x == y or y == '0'))
        more_general_parts.append(mg)
    return all(more_general_parts)

def fulfills(example, hypothesis):
    # The implementation is the same as for hypotheses:
    return more_general(hypothesis, example)

def min_generalizations(h, x):
    h_new = list(h)
    for i in range(len(h)):
        if not fulfills(x[i:i+1], h[i:i+1]):
            h_new[i] = '?' if h[i] != '0' else x[i]
    return [tuple(h_new)]

def min_specializations(h, domains, x):
    results = []
    for i in range(len(h)):
        if h[i] == '?':
            for val in domains[i]:
                if x[i] != val:
                    h_new = h[:i] + (val,) + h[i+1:]
                    results.append(h_new)
        elif h[i] != '0':
            h_new = h[:i] + ('0',) + h[i+1:]
            results.append(h_new)
    return results

with open('enjoysport.csv') as csvFile:
    examples = [tuple(line) for line in csv.reader(csvFile)]
```
Machine Learning Laboratory

Deepak D, Assistant Professor, Dept. of CS&E, Canara Engineering College, Mangaluru

```python
def get_domains(examples):
    d = [set() for i in examples[0]]
    for x in examples:
        for i, xi in enumerate(x):
            d[i].add(xi)
    return [list(sorted(x)) for x in d]

def specialize_G(x, domains, G, S):
    G_prev = list(G)
    for g in G_prev:
        if g not in G:
            continue
        if fulfills(x, g):
            G.remove(g)
            Gminus = min_specializations(g, domains, x)
    return G

def candidate_elimination(examples):
    domains = get_domains(examples)[:1]
    G = set([g_0(len(domains))])
    S = set([s_0(len(domains))])
    ...
i=0
print("\n G[{0}]:".format(i),G)
print("\n S[{0}]:".format(i),S)
for instance in examples:
    i=i+1
    x, label = instance[:-1], instance[-1]  # Splitting data into attributes and decisions
    if label=='yes': # x is positive example
        G = {g for g in G if fulfills(x, g)}
        S = generalize_S(x, G, S)
    else: # x is negative example
        S = {s for s in S if not fulfills(x, s)}
        G = specialize_G(x, domains, G, S)
print("\n G[{0}]:".format(i),G)
print("\n S[{0}]:".format(i),S)
return

candidate_elimination(examples)
Data Set:

<table>
<thead>
<tr>
<th></th>
<th>sunny</th>
<th>warm</th>
<th>normal</th>
<th>strong</th>
<th>warm</th>
<th>same</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>G3</td>
<td>'sunny'</td>
<td>'warm'</td>
<td>'normal'</td>
<td>'strong'</td>
<td>'warm'</td>
<td>'same'</td>
<td>yes</td>
</tr>
<tr>
<td>G4</td>
<td>'sunny'</td>
<td>'warm'</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Output:

G[0]: {('?', '?', '?', '?', '?', '?')}
S[0]: {('0', '0', '0', '0', '0', '0')}
G[1]: {('?', '?', '?', '?', '?', '?')}
S[1]: {('sunny', 'warm', 'normal', 'strong', 'warm', 'same')}
G[2]: {('?', '?', '?', '?', '?', '?')}
S[2]: {('sunny', 'warm', '?', 'strong', 'warm', 'same')}
G[3]: {('sunny', '?', '?', '?', '?', '?'), ('?', 'warm', '?', '?', '?', '?'), ('?', '?', '?', '?', '?', 'same')}
S[3]: {('sunny', 'warm', '?', 'strong', 'warm', 'same')}
G[4]: {('sunny', '?', '?', '?', '?', '?'), ('?', 'warm', '?', '?', '?', '?')}
S[4]: {('sunny', 'warm', '?', 'strong', '?', '?')}