```
import csv
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 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
def generalize_S(x, G, S):
        S_prev = list(S)
        for s in S_prev:
        if s not in S:
                continue
        if not fulfills(x, s):
                S.remove(s)
                Splus = min_generalizations(s, x)
        ## keep only generalizations that have a counterpart in G
                S.update([h for h in Splus if any([more_general(g,h) for g in G])])
        ## remove hypotheses less specific than any other in S
                S.difference_update([h for h in S if any([more_general(h, h1) for h1 in S if h != h1])])
        return S
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)
        ## keep only specializations that have a conuterpart in S
        G.update([h for h in Gminus if any([more_general(h, s) for s in S])])
        ## remove hypotheses less general than any other in G
        G.difference_update([h for h in G if any([more_general(g1, h) for g1 in G if h != g1])])
        return G
def candidate_elimination(examples):
```

```
    domains = get_domains(examples)[:-1]
    n = len(domains)
    G = set([("?",)*n])
    S = set([("0",)*n])
    print("Maximally specific hypotheses - S ")
    print("Maximally general hypotheses - G ")
    i=0
    print("\nS[0]:",str(S),"\nG[0]:",str(G))
    for xcx in examples:
    i=i+1
    x, cx = xcx[:-1], xcx[-1] # Splitting data into attributes and decisions
    if cx=='Y': # 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("\nS[{0}]:".format(i),S)
    print("G[{0}]:".format(i),G)
    return
with open('data22_sports.csv') as csvFile:
examples = [tuple(line) for line in csv.reader(csvFile)]
candidate_elimination(examples)
```