Code Jam 2011 - Qualification Round

Analysis: Magicka

This problem can be solved with a simulation. First, we have to remember what elements combine to make other elements. A map of some sort, like a hash map, is a great way of doing this. Next we have to track the opposed elements, remembering that one element can be opposed to multiple other elements; a set of pairs, while not particularly efficient for this purpose, will do the trick.

Finally, the simulation itself. For each character, first we check to see if it combines with the last item on the element list, and combine it if so. If it doesn't combine, then we iterate through the elements already in the list and see if it's opposed to any of them -- if so, we clear the list. Finally, if neither of those conditions was met, we append it to the list. Here is some Pythonesque pseudocode that solves the problem:

```
# Let combo_list contain all the combinations as 3-letter strs.
# Let opposed_list contain all the opposed elements as 2-letter strs.
# Let invoke be a str containing the elements to invoke.
combos = dict()
opposed = dict()
for x in combo_list:
  combos[x[0] + x[1]] = x[2]
  combos[x[1] + x[0]] = x[2]
for x in opposed_list:
  opposed.add(x[0] + x[1])
  opposed.add(x[1] + x[0])
# Now combos contains a mapping from each pair to the thing it
# creates.  If one of the combinations was "ABC", then
# combos["AB"] = "C" and combos["BA"] = "C".
# opposed is filled in a similar way.

element_list = []
for element in invoke:
  # If element_list isn't empty, the last element might combine
  # with the element being invoked.
  if element_list:
    last_two = element_list[-1] + element
    if last_two in combos:
      element_list[-1] = combos[last_two]
    continue

  # Now we iterate through element_list to see if anything there
  # is opposed to the element being invoked.
  wipe_list = False
  for e in element_list:
    if (e + element) in opposed:
      wipe_list = True
  if wipe_list:
    element_list = []
    continue

  # There was no combination and no erasing: just append the
  # element to the list.
  element_list.append(element)
```