Possible Solution for Parsing/Cleaning Data Exercise

Question 1

```
def get_year(input_list):

      output_list = []

      for lines in input_list:
            first_portion = lines.split("(")
            second_portion = first_portion[1].split(")")
            output_list.append(second_portion[0])

      return output_list
```

Question 2

```
def upper_case_cse(input_str):

    first_iteration = input_str.replace('c','C')
    second_iteration = first_iteration.replace('s','S')
    third_iteration = second_iteration.replace('e','E')

    return third_iteration
```

Possible Solution for Debugging Exercise

```
def parse_data():
      file = open("movie_genres.txt","r")
      movie_name = []
      movie_year = []
      movie_genre = []

      for line in file:
            line_data = line.split('" (')
            movie_name.append( line_data[0].replace('"','') )
            sub_line_data = line_data[1].split(')')
            movie_year.append(int(\
              sub_line_data[0].replace('/','').\
              replace('I','').\
              replace('V','').\
              replace('?','9')) )

            movie_genre.append(sub_line_data[1].
              replace('\t','').\
              replace('\n','') )


parse_data()
```

Possible Solution for Function Decomposition Exercise

Assume that the output of the function parse_data_to_dicts() returns a list of dictionaries with keys of Name, Year, and Genre.

1. Find a year's major category.

```
list_of_movies = parse_data_to_dicts()

parse_year_genre_data_by_key(list_of_movies, key, value)
      input: the data (list of dictionaries), string of key ('Year' or
            'Genre'), and key value (2010 or 'Action')
      output: list of respective data that matches key and value

      ex: parse_year_genre_data_by_key(list_of_movies, 'Year', 2010)
         will give you a list of movie genre in year 2010.

find_major_category(list)
      input: list of data
      output: the value which appears most often in the list
```

Use parse_year_genre_data_by_key with a specific year and the key 'Year' to get a list of all genres in the given year.

Use the list that is returned by parse_year_genre_data_by_key as an input for the function find_major_category which will return the genre that appears the most often.

2. Find the best (most productive) year for a certain category.

Reuse the parse_year_genre_data_by_key as above, with different inputs.

ex: parse_year_genre_data_by_key(list_of_movies, ‘Genre’, ‘Action’)
will give you a list of years in which action movies were created.

Use parse_year_genre_data_by_key to get a list of years in which the given genre was created.

Use the list that is returned by parse_year_genre_data_by_key as an input for the function find_major_category which will return the year that appears the most often.

Note: The problems above could have been solved by creating two functions, for example get_most_popular_genre(year) and get_most_popular_year(genre). But creating one function (like parse_year_genre_data_by_key) which instead takes a key like “Genre” or “Year” to do similar computations in both cases is better stylistically.