Project Final Submission Template

Step 1a: Planning

Identify the information in the file your program will read

Double click this cell to edit.

Describe (all) the information that is available. Be sure to note any surprising or unusual features. (For example, some information sources have missing data, which may be blank or flagged using values like -99, NaN, or something else.)

Step 1b: Planning

Brainstorm ideas for what your program will produce

Select the idea you will build on for subsequent steps

Double click this cell to edit.

You must brainstorm at least three ideas for graphs or charts that your program could produce and choose the one that you'd like to work on. You can choose between a line chart, histogram, bar chart, scatterplot, or pie chart.

Step 1c: Planning

Write or draw examples of what your program will produce

Double click this cell to edit.

You must include an image that shows what your chart or plot will look like. You can insert an image using the Insert Image command near the bottom of the Edit menu.

Step 2a: Building

Document which information you will represent in your data definitions

Design data definitions

Double click this cell to edit.

Before you design data definitions in the code cell below, you must explicitly document here which information in the file you chose to represent and why that information is crucial to the chart or graph that you'll produce when you complete step 2c.

In [30]:

```
from cs103 import *
from typing import NamedTuple, List
import csv
import matplotlib.pyplot as plt


##################
# Data Definitions

GlobalEcologicalFootprint = NamedTuple("GlobalEcologicalFootprint", [("country", str),
                                                                     ("region", str),
                                                                     ("total_footprint", float)]) #in range[0,
...)
# interp. the global ecological footprint for each region of the world based on the total ecological footprint (t
otal_footprint) in gha per person that
# the countries in each region produce. In order to solve for the average of the total ecological footrpint for t
he countries in each region, we need only
# country, region and total ecological footprint for this program.

GEF1 = GlobalEcologicalFootprint("Afghanistan", "Middle East/Central Asia", 0.79)
GEF2 = GlobalEcologicalFootprint("Albania", "Northern/Eastern Europe", 2.21)
GEF3 = GlobalEcologicalFootprint("Algeria", "Africa", 2.12)
GEF4 = GlobalEcologicalFootprint("Antigua and Barbuda", "Latin America", 5.38)
GEF5 = GlobalEcologicalFootprint("Australia", "Asia-Pacific", 9.31)
GEF6 = GlobalEcologicalFootprint("Austria", "European Union", 6.06)
GEF7 = GlobalEcologicalFootprint("Bermuda", "North America", 5.77)
GEF8 = GlobalEcologicalFootprint("Armenia", "Middle East/Central Asia", 2.23)
GEF9 = GlobalEcologicalFootprint("Belarus", "Northern/Eastern Europe", 5.09)
GEF10 = GlobalEcologicalFootprint("Angola", "Africa", 0.93)
GEF11 = GlobalEcologicalFootprint("Argentina", "Latin America", 3.14)
GEF12 = GlobalEcologicalFootprint("Bangladesh", "Asia-Pacific", 0.72)
GEF13 = GlobalEcologicalFootprint("Belgium", "European Union", 7.44)
GEF14 = GlobalEcologicalFootprint("Canada", "North America", 8.17)

@typecheck
def fn_for_global_ecological_footprint(gef: GlobalEcologicalFootprint) -> ...: #template based on Compound
    return ...(gef.country,
               gef.region,
               gef.total_footprint)


# List[GlobalEcologicalFootprint]
# interp. a list of countries based on the global ecological footprint

LOGEF0 = []
LOGEF1 = [GEF1, GEF2]
LOGEF2 = [GEF1, GEF2, GEF3]

# template based on arbitrary-sized data and the reference rule
@typecheck
def fn_for_logef(logef: List[GlobalEcologicalFootprint]) -> ...:
    #description of acc
    acc = ... #type: ...
    for gef in logef:
        acc = ...(acc, fn_for_global_ecological_footprint(gef))
    return acc
```

Step 2b and 2c: Building

Design a function to read the information and store it as data in your program

Design functions to analyze the data

Complete these steps in the code cell below. You will likely want to rename the analyze function so that the function name describes what your analysis function does.

In [36]:

```
###### # Functions

@typecheck
def main(filename: str) -> None:
    """
    Reads the file from given filename, analyzes the data, returns the result
    Returns a list of all the averages of all 7 regions, where (arbritrarily)
    first region: Middle East/Central Asia
    second region: Northern Eastern Europe
    third region: Africa
    fourth region: Latin America
```

```
    fifth region: Asia-Pacific
    sixth region: European Union
    seventh region: North America

    """
    # Template from HtDAP, based on function composition
    return fn_for_bar_chart(read(filename))


@typecheck
def read(filename: str) -> List[GlobalEcologicalFootprint]:
    """
    reads information from the specified file and returns a list of global ecological footprint
    """
    #return []  #stub
    # Template from HtDAP
    # logef contains the result so far
    logef = [] # type: List[GlobalEcologicalFootprint]

    with open(filename) as csvfile:

        reader = csv.reader(csvfile)
        next(reader) # skip header line

        for row in reader:
            # you may not need to store all the rows, and you may need
            # to convert some of the strings to other types
            gef = GlobalEcologicalFootprint(row[0], row[1] ,(parse_float(row[10])))
            logef.append(gef)

    return logef

# Begin testing
start_testing()

# Examples and tests for read
expect(read("empty_test.csv"), [])
expect(read("global_ecological_footprint_2016_UBC_test1.csv"), [GEF1, GEF2])
expect(read("global_ecological_footprint_2016_UBC_test2.csv"), [GlobalEcologicalFootprint("Antigua and Barbuda",
"Latin America", 5.38),
                                                                GlobalEcologicalFootprint("Argentina", "Latin Ame
rica", 3.14),
                                                                GlobalEcologicalFootprint("Armenia", "Middle East
/Central Asia", 2.23)])

# show testing summary
summary()

@typecheck
def fn_for_bar_chart(logef: List[GlobalEcologicalFootprint]) -> None:
    """
    Returns a bar chart using the averages of the total ecological footprint from a list of
    countries based on the global ecological footprint
    """
    #return None #stub
    #template based on visualization

    list_of_avgs = [avg_first_region(logef), avg_second_region(logef), avg_third_region(logef), avg_fourth_region
(logef), avg_fifth_region(logef), avg_sixth_region(logef), avg_seventh_region(logef)]

    plt.xlabel('Region Number')
    plt.ylabel('Average of Total Ecological Footprint (gha/person)')
    plt.title('Average of Total Ecological Footprint by Region')

    bar_width = 20

    plt.bar(1, avg_first_region(logef), label = "Middle East/Central Asia")
    plt.bar(2, avg_second_region(logef), label = "Northern Eastern Europe")
    plt.bar(3, avg_third_region(logef), label = "Africa")
    plt.bar(4, avg_fourth_region(logef), label = "Latin America")
    plt.bar(5, avg_fifth_region(logef), label = "Asia-Pacific")
    plt.bar(6, avg_sixth_region(logef), label = "European Union")
    plt.bar(7, avg_seventh_region(logef), label = "North America")

    plt.legend()

    plt.show()

    return None

@typecheck
def avg_seventh_region(logef: List[GlobalEcologicalFootprint]) -> float:
    """
```

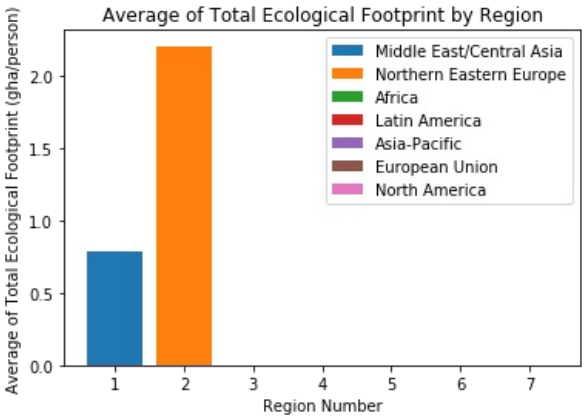
```
    Returns the average of the total ecological footprint for a list of countries based on the global ecological
footprint
    only for North America
    """
    #return 5.0 #stub
    return average_total_footprint(filter_seventh_region(logef))

@typecheck
def filter_seventh_region(logef: List[GlobalEcologicalFootprint]) -> List[GlobalEcologicalFootprint]:
    """
    Returns a list of global ecological footprint with only North America region
    """
    #return True #stub
    #template based on List[GlobalEcologicalFootprint]
    acc = [] #type: int
    for gef in logef:
        if seventh_region(gef) is True:
            acc.append(gef)
    return acc

@typecheck
def seventh_region(gef: GlobalEcologicalFootprint) -> bool:
    """
    Returns True if the region is North America
    """
    #return True #stub
    #template based on GlobalEcologicalFootprint
    return gef.region == "North America"

@typecheck
def avg_sixth_region(logef: List[GlobalEcologicalFootprint]) -> float:
    """
    Returns the average of the total ecological footprint for a list of countries based on the global ecological
footprint
    only for European Union
    """
    #return 5.0 #stub
    return average_total_footprint(filter_sixth_region(logef))

@typecheck
def filter_sixth_region(logef: List[GlobalEcologicalFootprint]) -> List[GlobalEcologicalFootprint]:
    """
    Returns a list of global ecological footprint with only European Union region
    """
    #return True #stub
    #template based on List[GlobalEcologicalFootprint]
    acc = [] #type: int
    for gef in logef:
        if sixth_region(gef) is True:
            acc.append(gef)
    return acc

@typecheck
def sixth_region(gef: GlobalEcologicalFootprint) -> bool:
    """
    Returns True if the region is European Union
    """
    #return True #stub
    #template based on GlobalEcologicalFootprint
    return gef.region == "European Union"

@typecheck
def avg_fifth_region(logef: List[GlobalEcologicalFootprint]) -> float:
    """
    Returns the average of the total ecological footprint for a list of countries based on the global ecological
footprint
    only for Asia-Pacific
    """
    #return 5.0 #stub
    return average_total_footprint(filter_fifth_region(logef))

@typecheck
def filter_fifth_region(logef: List[GlobalEcologicalFootprint]) -> List[GlobalEcologicalFootprint]:
    """
    Returns a list of global ecological footprint with only Asia-Pacific region
    """
    #return True #stub
    #template based on List[GlobalEcologicalFootprint]
    acc = [] #type: int
    for gef in logef:
        if fifth_region(gef) is True:
            acc.append(gef)
    return acc
```

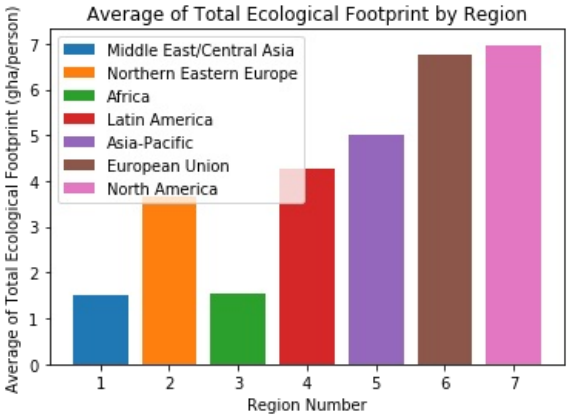
```python
@typecheck
def fifth_region(gef: GlobalEcologicalFootprint) -> bool:
    """
    Returns True if the region is Asia-Pacific
    """
    #return True #stub
    #template based on GlobalEcologicalFootprint
    return gef.region == "Asia-Pacific"

@typecheck
def avg_fourth_region(logef: List[GlobalEcologicalFootprint]) -> float:
    """
    Returns the average of the total ecological footprint for a list of countries based on the global ecological
footprint
    only for Latin America
    """
    #return 5.0 #stub
    return average_total_footprint(filter_fourth_region(logef))

@typecheck
def filter_fourth_region(logef: List[GlobalEcologicalFootprint]) -> List[GlobalEcologicalFootprint]:
    """
    Returns a list of global ecological footprint with only Latin America region
    """
    #return True #stub
    #template based on List[GlobalEcologicalFootprint]
    acc = [] #type: int
    for gef in logef:
        if fourth_region(gef) is True:
            acc.append(gef)
    return acc

@typecheck
def fourth_region(gef: GlobalEcologicalFootprint) -> bool:
    """
    Returns True if the region is Latin America
    """
    #return True #stub
    #template based on GlobalEcologicalFootprint
    return gef.region == "Latin America"

@typecheck
def avg_third_region(logef: List[GlobalEcologicalFootprint]) -> float:
    """
    Returns the average of the total ecological footprint for a list of countries based on the global ecological
footprint
    only for Africa
    """
    #return 5.0 #stub
    return average_total_footprint(filter_third_region(logef))

@typecheck
def filter_third_region(logef: List[GlobalEcologicalFootprint]) -> List[GlobalEcologicalFootprint]:
    """
    Returns a list of global ecological footprint with only Africa region
    """
    #return True #stub
    #template based on List[GlobalEcologicalFootprint]
    acc = [] #type: int
    for gef in logef:
        if third_region(gef) is True:
            acc.append(gef)
    return acc

@typecheck
def third_region(gef: GlobalEcologicalFootprint) -> bool:
    """
    Returns True if the region is Africa
    """
    #return True #stub
    #template based on GlobalEcologicalFootprint
    return gef.region == "Africa"

@typecheck
def avg_second_region(logef: List[GlobalEcologicalFootprint]) -> float:
    """
    Returns the average of the total ecological footprint for a list of countries based on the global ecological
footprint
    only for Northern/Eastern Europe
    """
    #return 5.0 #stub
    return average_total_footprint(filter_second_region(logef))
```

```
@typecheck
def filter_second_region(logef: List[GlobalEcologicalFootprint]) -> List[GlobalEcologicalFootprint]:
    """
    Returns a list of global ecological footprint with only Northern/Eastern Europe region
    """
    #return True #stub
    #template based on List[GlobalEcologicalFootprint]
    acc = [] #type: int
    for gef in logef:
        if second_region(gef) is True:
            acc.append(gef)
    return acc

@typecheck
def second_region(gef: GlobalEcologicalFootprint) -> bool:
    """
    Returns True if the region is Northern/Eastern Europe
    """
    #return True #stub
    #template based on GlobalEcologicalFootprint
    return gef.region == "Northern/Eastern Europe"


@typecheck
def avg_first_region(logef: List[GlobalEcologicalFootprint]) -> float:
    """
    Returns the average of the total ecological footprint for a list of countries based on the global ecological
footprint
    only for Middle East/Central Asia
    """
    #return 5.0 #stub
    return average_total_footprint(filter_first_region(logef))

@typecheck
def filter_first_region(logef: List[GlobalEcologicalFootprint]) -> List[GlobalEcologicalFootprint]:
    """
    Returns a list of global ecological footprint with only Middle East/Central Asia region
    """
    #return True #stub
    #template based on List[GlobalEcologicalFootprint]
    acc = [] #type: List[GlobalEcologicalFootprint]
    for gef in logef:
        if first_region(gef) is True:
            acc.append(gef)
    return acc

@typecheck
def first_region(gef: GlobalEcologicalFootprint) -> bool:
    """
    Returns True if the region is Middle East/Central Asia
    """
    #return True #stub
    #template based on GlobalEcologicalFootprint
    return gef.region == "Middle East/Central Asia"

@typecheck
def average_total_footprint(logef: List[GlobalEcologicalFootprint]) -> float:
    """
    Returns the average of the total ecological footprint for a list of countries based on the global ecological
footprint
    """
    #return 5.0 #stub
    #template based on List[GlobalEcologicalFootprint]
    #acc contains the results so far
    acc = 0 #type: float
    if len(logef) == 0:
        return 0 #this is so that we don't divide by 0
    else:
        for gef in logef:
            acc = acc + gef.total_footprint
        return acc/(len(logef))


start_testing()

expect(fn_for_bar_chart([GEF1, GEF2]), None)
expect(fn_for_bar_chart([GEF1, GEF2, GEF3, GEF4, GEF5, GEF6, GEF7, GEF8, GEF9, GEF10, GEF11, GEF12, GEF13, GEF14]
), None)

expect(avg_seventh_region([GEF1, GEF7]), 5.77)
expect(avg_seventh_region([GEF1, GEF7, GEF14]), 6.97)
```

```
expect(filter_seventh_region([GEF1, GEF7]), [GEF7])

expect(filter_seventh_region([GEF1, GEF7, GEF14]), [GEF7, GEF14])

expect(seventh_region(GEF1), False)
expect(seventh_region(GEF7), True)

expect(avg_sixth_region([GEF1, GEF6]), 6.06)
expect(avg_sixth_region([GEF1, GEF6, GEF13]), 6.75)

expect(filter_sixth_region([GEF1, GEF6]), [GEF6])
expect(filter_sixth_region([GEF1, GEF6, GEF13]), [GEF6, GEF13])

expect(sixth_region(GEF1), False)
expect(sixth_region(GEF6), True)

expect(avg_fifth_region([GEF1, GEF5]), 9.31)
expect(avg_fifth_region([GEF1, GEF5, GEF12]), 5.015)

expect(filter_fifth_region([GEF1, GEF5]), [GEF5])
expect(filter_fifth_region([GEF1, GEF5, GEF12]), [GEF5, GEF12])

expect(fifth_region(GEF1), False)
expect(fifth_region(GEF5), True)

expect(avg_fourth_region([GEF1, GEF4]), 5.38)
expect(avg_fourth_region([GEF1, GEF4, GEF11]), 4.26)

expect(filter_fourth_region([GEF1, GEF4]), [GEF4])
expect(filter_fourth_region([GEF1, GEF4, GEF11]), [GEF4, GEF11])

expect(fourth_region(GEF1), False)
expect(fourth_region(GEF4), True)

expect(avg_third_region([GEF1, GEF3]), 2.12)
expect(avg_third_region([GEF1, GEF3, GEF10]), 1.525)

expect(filter_third_region([GEF1, GEF3]), [GEF3])
expect(filter_third_region([GEF1, GEF3, GEF10]), [GEF3, GEF10])

expect(third_region(GEF1), False)
expect(third_region(GEF3), True)

expect(avg_second_region([GEF1, GEF2]), 2.21)
expect(avg_second_region([GEF1, GEF2, GEF9]), 3.65)

expect(filter_second_region([GEF1, GEF2]), [GEF2])
expect(filter_second_region([GEF1, GEF2, GEF9]), [GEF2, GEF9])

expect(second_region(GEF1), False)
expect(second_region(GEF2), True)

expect(avg_first_region([GEF1, GEF2]), 0.79)
expect(avg_first_region([GEF1, GEF2, GEF8]), 1.51)

expect(filter_first_region([GEF1, GEF2]), [GEF1])
expect(filter_first_region([GEF1, GEF2, GEF8]), [GEF1, GEF8])

expect(first_region(GEF1), True)
expect(first_region(GEF2), False)

expect(average_total_footprint([]), 0)
expect(average_total_footprint([GEF1, GEF2]), 1.5)
summary()
```

```
3 of 3 tests passed
```





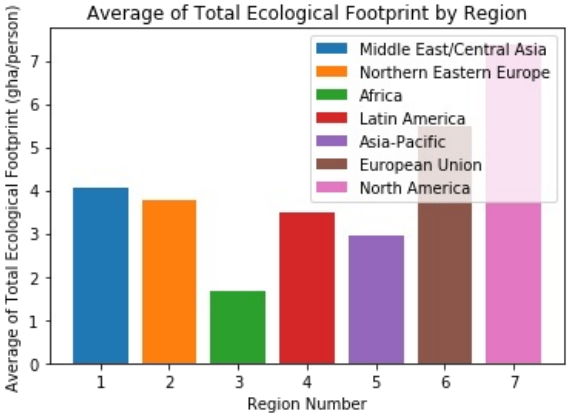

```
46 of 46 tests passed
```

Final Graph/Chart

Now that everything is working, you **must** call `main` on the intended information source in order to display the final graph/chart:

In [37]:

```
main("global_ecological_footprint_2016_UBC.csv")
```



In [ ]:

```
# Be sure to select ALL THE FILES YOU NEED (including csv's)
# when you submit. As usual, you cannot edit this cell.
# Instead, run this cell to start the submission process.
from cs103 import submit

COURSE = 35980
ASSIGNMENT = 420474  # final submission
submit(COURSE, ASSIGNMENT)

# If your submission fails, SUBMIT ANYWAY by downloading your files and uploading them to Canvas.
# You can learn how on the page "How to submit your Jupyter notebook" on our Canvas site.
```