

# 1 Introduction

This project is on creating an interactive database searching and reporting program. As it states in the syllabus.

**Projects are to be done strictly on your own and as with all assignments the sharing of files and code is strictly prohibited and constitutes an act of Academic Misconduct. Furthermore the use of any electronic medium, such as code repositories, forums, blogs, message boards, email, etc. is strictly prohibited and constitutes an act of Academic Misconduct.**

The only person you may discuss this with in any form is me. You may use the textbook, the textbook example code, and the class example code that is posted on the MyClasses site.

When you are ready to submit your work create a folder called `Project02`. Put all the code files needed for the project in the folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire `Project02` folder up into a single zip file and submit it.

# 2 Life Expectancy Database Reporter

The file `LifeExpWorld.txt` contains data on the life expectancy of individuals given what year they were born, their gender, and where they live. The file is tab delimited, that is, the entries on one line represent one record of information and the items are separated by tab characters. The beginning of this file is listed below.

```
Country Code Country Name Region Income Group Gender Year Life Expectancy
ABW Aruba Latin America & Caribbean High income Male 1960 64.084
AFG Afghanistan South Asia Low income Male 1960 31.58
AGO Angola Sub-Saharan Africa Lower middle income Male 1960 31.787
ALB Albania Europe & Central Asia Upper middle income Male 1960 61.309
AND Andorra Europe & Central Asia High income Male 1960
ARB Arab World Male 1960 45.776879216
ARE United Arab Emirates Middle East & North Africa High income Male 1960 49.677
.
.
.
```

The first line are column headers. Do not remove this line from the file, instead, your program should read past this line before it begins processing the file. Each line of the file represents a record that contains life expectancy data for a person of a particular gender, date of birth, and country they live in. There is also data on the region and income group. In particular, the first item is a three character country code, then the country name, region, income group, gender, year (of birth), and finally the life expectancy of the person. Not all lines contain all of the data but they do all contain tabs between the fields it is just that some fields are empty. All records have a country code, country name, gender, and year of birth. Not all of them have a region, income group, or life expectancy entry. Note from the

data above that the life expectancy for a male born in Andorra in 1960 is not given and the same entry for the Arab World is missing both the region and income group.

Your program is to define a struct that will contain each of the fields for a record. It should read the entire file into either an array or a vector of these structs, your choice.

Then the program should do the following,

1. Go through the entire set of records and find all of the different gender types in the set. Then print them to the console and ask the user to select one of them. Selection will be an integer, simply number the gender types starting at 1. The gender names are to be sorted alphabetically. The program must check to make sure that the selection is valid.
2. The program is to then go through the data again and list all of the birth years associated with that gender. These are also to be sorted and of course each is to be listed only once. The program should then ask the user to select a year. The selection here is the actual year (such as 1976) and the program must check to make sure that the input year is one that is listed. Note that it is possible that some years are skipped.
3. The program is to then go through the data set again and extract all of the countries associated with the gender and year choices already made. This is to be displayed to the console in alphabetically sorted order and numbered from 1 on. There should not be any duplicate entries in this list, as with the previous two lists. The user is to then select a country by inputting its number. The program must check to make sure that the selection is valid.
4. Once all of the selections have been entered, the program should print out all of the records that match the year, gender, and country that were selected, there may be more than one. If no record exists then the program should display a message that there were no records matching the selection made. For example,

```
Records Found
=====
Country:  United States (USA)
Region:   North America
Income Group:  High income
Gender:   Male
Year of Birth: 1965
Life Expectancy: 66.8 Years
```

If the region is not listed for that record then simply do not print it out, same goes for the income group. For example,

```
Records Found
=====
Country:  Arab World (ARB)
Gender:   Male
Year of Birth: 1960
Life Expectancy: 45.7769 Years
```

Also, if the life expectancy is not listed then have the program print out “None Given”. For example,

```
Records Found
=====
Country:  Andorra (AND)
Region:   Europe & Central Asia
Income Group: High income
Gender:   Male
Year of Birth: 1960
Life Expectancy: None Given
```

## 2.1 Hints & Notes

1. Each data record in the file is on a single line. There are tabs between the fields and there could be spaces in strings representing the country, region and/or income group. With this structure, streaming in each field could be difficult. It would probably be easier to read the entire line in, search for the tabs and use substrings to extract the fields.
2. Most of your fields will be strings but the year of birth and the life expectancy will obviously be numeric. If you have not used the `stoi`, `stof`, or `stod` functions from the string library you may wish to look those up. Note that if the argument of these is not a string representation of a valid number of the designated type the function will throw an exception (crash the program). This includes the empty string. You may also use `atoi` and `atof` if you wish, but for these the argument must be a C-String (remember the `.c_str()` method).
3. Do not hard code any of the options, such as gender and years, take them from the data. This program is to work with any file that has the same structure. So my test file may have different countries, additional years, missing years, genders, etc.
4. Do not assume that any of the data is in order, alphabetical or numeric. Again, my test file may have the entries all mixed up.
5. Since there will be some string processing, you may wish to copy the string functions that you created a while back into this project. If you had some difficulties with those you may use my solutions.

## 2.2 Example Run

A complete run of the program is below.

```
Select Gender Type
=====
1. Female
2. Male
3. Total

Selection: 2
```

Select Birth Year

=====

1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015

Selection: 1965

Select Country

=====

1. Afghanistan  
2. Albania  
3. Algeria  
4. American Samoa  
5. Andorra  
6. Angola  
7. Antigua and Barbuda

8. Arab World
9. Argentina
10. Armenia
11. Aruba
12. Australia
13. Austria
14. Azerbaijan
15. Bahamas, The
16. Bahrain
17. Bangladesh
18. Barbados
19. Belarus
20. Belgium
21. Belize
22. Benin
23. Bermuda
24. Bhutan
25. Bolivia
26. Bosnia and Herzegovina
27. Botswana
28. Brazil
29. British Virgin Islands
30. Brunei Darussalam
31. Bulgaria
32. Burkina Faso
33. Burundi
34. Cabo Verde
35. Cambodia
36. Cameroon
37. Canada
38. Caribbean small states
39. Cayman Islands
40. Central African Republic
41. Central Europe and the Baltics
42. Chad
43. Channel Islands
44. Chile
45. China
46. Colombia
47. Comoros
48. Congo, Dem. Rep.
49. Congo, Rep.
50. Costa Rica
51. Cote d'Ivoire
52. Croatia
53. Cuba
54. Curacao
55. Cyprus
56. Czech Republic
57. Denmark
58. Djibouti
59. Dominica
60. Dominican Republic
61. Early-demographic dividend
62. East Asia & Pacific
63. East Asia & Pacific (IDA & IBRD countries)
64. East Asia & Pacific (excluding high income)
65. Ecuador
66. Egypt, Arab Rep.
67. El Salvador
68. Equatorial Guinea
69. Eritrea
70. Estonia
71. Ethiopia
72. Euro area
73. Europe & Central Asia
74. Europe & Central Asia (IDA & IBRD countries)
75. Europe & Central Asia (excluding high income)
76. European Union
77. Faroe Islands
78. Fiji

---

79. Finland  
80. Fragile and conflict affected situations  
81. France  
82. French Polynesia  
83. Gabon  
84. Gambia, The  
85. Georgia  
86. Germany  
87. Ghana  
88. Gibraltar  
89. Greece  
90. Greenland  
91. Grenada  
92. Guam  
93. Guatemala  
94. Guinea  
95. Guinea-Bissau  
96. Guyana  
97. Haiti  
98. Heavily indebted poor countries (HIPC)  
99. High income  
100. Honduras  
101. Hong Kong SAR, China  
102. Hungary  
103. IBRD only  
104. IDA & IBRD total  
105. IDA blend  
106. IDA only  
107. IDA total  
108. Iceland  
109. India  
110. Indonesia  
111. Iran, Islamic Rep.  
112. Iraq  
113. Ireland  
114. Isle of Man  
115. Israel  
116. Italy  
117. Jamaica  
118. Japan  
119. Jordan  
120. Kazakhstan  
121. Kenya  
122. Kiribati  
123. Korea, Dem. People's Rep.  
124. Korea, Rep.  
125. Kosovo  
126. Kuwait  
127. Kyrgyz Republic  
128. Lao PDR  
129. Late-demographic dividend  
130. Latin America & Caribbean  
131. Latin America & Caribbean (excluding high income)  
132. Latin America & the Caribbean (IDA & IBRD countries)  
133. Latvia  
134. Least developed countries: UN classification  
135. Lebanon  
136. Lesotho  
137. Liberia  
138. Libya  
139. Liechtenstein  
140. Lithuania  
141. Low & middle income  
142. Low income  
143. Lower middle income  
144. Luxembourg  
145. Macao SAR, China  
146. Macedonia, FYR  
147. Madagascar  
148. Malawi  
149. Malaysia

---

---

150. Maldives  
151. Mali  
152. Malta  
153. Marshall Islands  
154. Mauritania  
155. Mauritius  
156. Mexico  
157. Micronesia, Fed. Sts.  
158. Middle East & North Africa  
159. Middle East & North Africa (IDA & IBRD countries)  
160. Middle East & North Africa (excluding high income)  
161. Middle income  
162. Moldova  
163. Monaco  
164. Mongolia  
165. Montenegro  
166. Morocco  
167. Mozambique  
168. Myanmar  
169. Namibia  
170. Nauru  
171. Nepal  
172. Netherlands  
173. New Caledonia  
174. New Zealand  
175. Nicaragua  
176. Niger  
177. Nigeria  
178. North America  
179. Northern Mariana Islands  
180. Norway  
181. Not classified  
182. OECD members  
183. Oman  
184. Other small states  
185. Pacific island small states  
186. Pakistan  
187. Palau  
188. Panama  
189. Papua New Guinea  
190. Paraguay  
191. Peru  
192. Philippines  
193. Poland  
194. Portugal  
195. Post-demographic dividend  
196. Pre-demographic dividend  
197. Puerto Rico  
198. Qatar  
199. Romania  
200. Russian Federation  
201. Rwanda  
202. Samoa  
203. San Marino  
204. Sao Tome and Principe  
205. Saudi Arabia  
206. Senegal  
207. Serbia  
208. Seychelles  
209. Sierra Leone  
210. Singapore  
211. Sint Maarten (Dutch part)  
212. Slovak Republic  
213. Slovenia  
214. Small states  
215. Solomon Islands  
216. Somalia  
217. South Africa  
218. South Asia  
219. South Asia (IDA & IBRD)  
220. South Sudan

---

```
221. Spain
222. Sri Lanka
223. St. Kitts and Nevis
224. St. Lucia
225. St. Martin (French part)
226. St. Vincent and the Grenadines
227. Sub-Saharan Africa
228. Sub-Saharan Africa (IDA & IBRD countries)
229. Sub-Saharan Africa (excluding high income)
230. Sudan
231. Suriname
232. Swaziland
233. Sweden
234. Switzerland
235. Syrian Arab Republic
236. Tajikistan
237. Tanzania
238. Thailand
239. Timor-Leste
240. Togo
241. Tonga
242. Trinidad and Tobago
243. Tunisia
244. Turkey
245. Turkmenistan
246. Turks and Caicos Islands
247. Tuvalu
248. Uganda
249. Ukraine
250. United Arab Emirates
251. United Kingdom
252. United States
253. Upper middle income
254. Uruguay
255. Uzbekistan
256. Vanuatu
257. Venezuela, RB
258. Vietnam
259. Virgin Islands (U.S.)
260. West Bank and Gaza
261. World
262. Yemen, Rep.
263. Zambia
264. Zimbabwe
```

```
Selection: 252
```

```
Records Found
```

```
=====
```

```
Country:  United States (USA)
```

```
Region:  North America
```

```
Income Group:  High income
```

```
Gender:  Male
```

```
Year of Birth:  1965
```

```
Life Expectancy:  66.8 Years
```

### 3 Grading

The program itself should, of course, be nicely formatted and commented and should follow all the other rules of good programming style. Make sure you are following all the coding and documentation standards of the class that are published on the MyClasses site for this class.

The grading of the project will take two forms, a sample run and an inspection of the



code. If the program does not run you will receive a zero for that portion. So even if the program is not complete you will get a better grade for a partial program that runs verses a program that does not run. So I would suggest a completion in stages approach. The run portion of the grading will test the user interface for usability and conforming to the specifications I have outlined above. The code inspection portion of the grade will involve commenting, readability, correct indentation, variable names, structure and style, correctness, and conforming to specifications.

I am looking for clean, easy to read, code. A good use of functions without overuse. Commenting that gets the point across concisely. An easy to use interface with nice looking output.