Henry Tran
Lee Chang
IS 520
Gove Allen

Around the World in a Bajillion Loops - A Tale of an IS Final

IS 520 Final Project Write-up

Introduction & Executive Summary - Lee

Using online ticket finding webpages can be a hassle. Annoying advertisements harass the user. Only one search destination is allowed at a time. Lots of scrolling, re-inputting information is required. We created a program that makes vacation planning easier by cutting down the unnecessary interactions the user has with websites and provides the same information in a streamlined manner. The project has two portions:

1. Airfare

   The first portion provides a user input form to allow a user to quickly compare up to five routes that they would like to travel to. The program finds the price range, and the span of airlines that service that route. It utilizes a large range of arrays to do so. (**range** of **arrays**, get the word play? Had to use it!)

2. Living Costs

   Each country's living standards and daily expenses are different. For a user with a budget, knowing the real time cost of staying in a country is important. This portion of the project allows the user to input a country of interest and returns the cost of living there. The results includes information such as housing cost, food cost, transportation cost, entertainment costs, etc.

The following sections will break down the code structure of the program and explain the functions, procedures, methods, etc. that we used to accomplish this project.

I. Airfare - Lee

This portion relates to the airfare and will be broken down in the following sections:

a. "Flight Information" User Form
   a. Buttons
b. Data Scrape
   a. Airport Code Directory
   b. Ticket Prices
   c. Miscellaneous Notes
c. Summary of Codes

<u>"Flight Information" User Form:</u>

The "Flight Information" User Form contains the following:

1. A combo-box that determines which route's information is being edited
2. A combo-box that allows the user to select the max number of stops in the flight they would like.

Henry Tran
Lee Chang
IS 520
Gove Allen

3. Departure and destination locations combo-boxes which are populated with airport location and codes. These codes and locations come from a sheet called “Airport Code Directory” which in turn, scraped the information from the internet.
4. Departure and Return Dates textboxes.
5. 4 buttons: “Clear” “Reset” “Cancel” “Get Info”

Information the user inputs and the information outputted by the program is displayed primarily on a worksheets named “Main Page.” The user can edit up to five routes. The input and output data for the five routes are simultaneously viewable on the “Main Page”

“Flight Information” Initialization: The “Flight Information” form is initialized when the user clicks the “Input Flight Info” button.

1) Populating the Directory - The first thing the user form does on initialization is check to see if the Airport Code Directory is populated. The Directory should always fill in Range “A1.” If that range is not filled, then the form will call the sub procedure to populate that sheet. This portion is crucial because the combo boxes for Departure location and Destination location must be populated based off of the Airport Codes page by using the “With…AddItem…End With” methods.
    a. Loops: the AddItem methods are used within an “If…Then….End If” to only read cells which are not empty. This If Then statement is then contained within a “For” Loop which reads runs through the bottom of the airport codes list.
    b. Appending the Code: the procedure also grabs the code at accompanying the city and appends the code to the end of the city/country name

2) COMBO1: Once the user form as confirmed that the Directory has been populated, fills in fields through the following options:
    a. If the active cell is within rows 2 and 6, then the user form will use the information in that row to populate the form’s fields.
    b. If the user has selected a cell outside of the range, then the user form will automatically use row 2, the first row with information, to populate the user form.

    Both methods of populating the form are done with a loop that selects cell (row, 2) in the respective row and offsetting by (0, 1) to move across the information fields.

    A key factor that begins affecting all other aspects of this workbook is the COMBO1 box, which is labeled as the “Trip No.” combo box. This combo box derives its value from the active row. Other procedures I used will base their input and output ranges using this combo box value. (The user can change the “Trip no” to edit a different row without exiting the user form).

3) Data Reading: The user form then reads the active row’s information by using the combo box as a row reference. The roundtrip checkbox corresponds to the cell with values “Yes” or “No.”

Henry Tran
Lee Chang
IS 520
Gove Allen

The dates, locations, and max number of stops are all read from the cells and inputted into the User Form for the User to manipulate. If the row is blank, then the form is blank as well.

Buttons

The User has a choice of four command buttons to use: "Get Info," "Cancel," "Clear," and "Reset." Each of these are functional and will be described.

"Get Info"

1) **Data Input:** when the user changes the information in the user form and clicks "Get Info," the form calls the **GetPrices.getTicketPrices sub** to get the information from the internet.
    a. **Data Checks and Validation:** when the user hits the "Get Info" button, the user form checks for the following:
        i. If the dates are the same
        ii. If either dates are earlier than today
        iii. If the return date is earlier than the departure date.
        iv. If the airport selected matches an airport on the Directory
        v. If any of the fields are left blank
    b. **Date formatting:** if the data is correct, then the form will format the data before putting it into the cells. When the dates are read or inputted into the worksheet, the "CDate" function must be used to preserve the traditional date format to prevent it from reverting into a number code.

"Reset"

This button simply reverts the form back to the original values of the row after initialization.

"Cancel"

This button uses "Unload" to prevent any changes from occurring.

"Clear"

This button clears all the fields in the "Flight Information" form.

Data Scrape

Airport codes

The airport codes are taken from world-airport-codes.com and consist of the top 40 domestic airports, and the top 50 international airports. Some of the airports overlap, and that is address in the code. The airport codes are in IATA format so that they can be inputted into our ticket price procedure with no issues. We only chose the busiest and most popular airports for this project to preserve project scale. The complete list of airports comprises of over 700 codes and locations.

Henry Tran
Lee Chang
IS 520
Gove Allen

1. Getting the Information

   The sub-procedure for getting the airport information is very simple, but repetitive. After dim "a" as an agent, and setting it as a new agent using Prof. Allen's agent code, the procedure opens the webpage sets **a.position = 1,** and uses **a.moveto** to move to various phrases in the web page code such as "<tr>," "<td>," and "<td/>." These are parts of the code unique to the portions on the webpage containing the airport name, location, and code.

   A **"Do while a.moveto….loop"** loop is used for the airport name, location, and code. The name loop, location loop, and code loop are all the same format, but separate and independent loops. Using the name loop as a model, the loop uses **"nameTemp = nameTemp & "|" & "TempVar**" combination to tack on each addition name onto "nameTemp" creating a long string of names such as the following:

   "John F. Kennedy|Los Angeles International| Sky Harbor|John Wayne International….."

   Then names are then turned into an **array using the Split function**, splitting the string by the "|" symbol. These names are then outputted onto the "Airport Code Directory" sheet accordingly. This model is repeated for the Location and Code.

   It is important to note that the entire section of code as described is used once for the webpage for domestic locations, and **completely repeated again**, but for the international locations. The international information is appended to the bottom of the domestic information using the following code structure (airportArray is the domestic array while nameArray is the international name array):

```
For i = UBound(AirportArray) To UBound(nameArray)
    Sheets("Airport Code Directory").Cells(i, 1).Value = "I- " &
    nameArray(i)
Next
```

2. Cleaning up the Directory

   The some of the codes are doubles, since the most traveled international airports list also included some on the Domestic list. The codes and related information are **sorted by using the following segment of code:**

```
ActiveWorkbook.ActiveSheet.sort.SortFields.Clear
ActiveSheet.Range("a1").Select
ActiveSheet.sort.SortFields.Add Key:=Range("C1",
Selection.End(xldown)), _
SortOn:=xlSortOnValues, Order:=xlAscending,
DataOption:=xlSortNormal
With ActiveSheet.sort
.SetRange ActiveCell.CurrentRegion
.Header = xlGuess
.MatchCase = False
.Orientation = xlTopToBottom
```

Henry Tran
Lee Chang
IS 520
Gove Allen

```
.SortMethod = xlPinYin
.Apply
End With
```

This code sorts the whole Directory by the airport codes so that similar codes are placed next to each other. Then I compared each airport code with the consecutive airport code. If the two airport codes were the same, then one of the double was eliminated. This process eliminated all doubles regardless of airport name discrepancies and would delete the entire row for the second occurrence of the airport code.

The Directory was then sorted by the first column alphabetically to prepare it to populate the "Flight Information" user form. This was purely for more convenience for the user to navigate the Departure and Destination locations.

A "D" was added to the front of all domestic locations and an "I" was added onto all International locations to make it easier for the user.

Ticket Prices

The ticket prices were found using the same agent and **a.moveto loop** models as the Directory. However, the website URL was modified each time the user manipulated the data. The sections of the code are detailed below.

1. **Opening the webpage:** the webpage URL was modified using a URLTemplate:

"https://www.expedia.com/Flights-Search?trip=roundtrip&leg1=from:<DEPARTCODE>,to:<DESTCODE>,departure:<DEPARTDATE>TANYT&leg2=from:<DESTCODE>,to:<DEPARTCODE>,departure:<RETURNDATE>TANYT&passengers=children:0,adults:1,seniors:0,infantinlap:Y&mode=search"

In which the segments within the various "< >" were replaced with the user inputted information saved in the cells of the worksheet (see "Flight Information" section) using the **Replace()** function.

2. **Data Gathering:** after the webpage is opened, the code uses the same model of **a.moveto** and **a.gettext** to grab the price, airline name, flight time, and number of stops. The same method of using an elongated tack-on string with a **Split()** function was used to turn the information into arrays and then put into a separated worksheet.
3. **Lack of Routes:** if the webpage declares no flights (such as between JFK and LaGuardia, then a message box will appear)
4. **Data Cleaning:** the same methods to sort and eliminate duplicates is used to take out the redundancies where the price, number of stops, and airlines are the same, regardless of flight time.
5. **Data Transfer:** Finally, the information is put into the "Main Page" where the user originally was. The procedure uses a combination of **Min/Max** functions to find the range of the prices. The code eliminates all results from the web page containing "Multiple Airlines" because these flights were

outrageously higher priced than the average, often three times more than the majority of flights. I eliminated them because I felt they skewed the data unfavorably.

6. Beep: The function Beeps when it is completed.

Miscellaneous

1) For some reason, if the computer does not populate the ranges to be sorted quickly enough, or the computer is too busy, an error will occur regarding the sort range being wrongly referenced. The User can either restart the sub, or manually drag the yellow arrow back to the beginning of the sub. I should have eliminated this problem by having the sub run one trip at a time as demanded by the user.
2) The User will see a "Reset Project" button which can be used to clear all the fields and start a fresh search
3) The user can see the detailed flight information by clicking the hyperlink embedded into each column of information.

Summary of Codes

This is a general summary of important functions, methods, etc. used in this portion of the project:

1. Do, Do while, Do until Loops
2. For…Next Loops
3. Len, Right, Left functions
4. A.moveto, a.gettext
5. With….End With
6. Dynamic Arrays
7. User Forms
8. Buttons
9. Sort Codes
10. If….Then….ElseIf….End If Statements
11. Split Functions
12. Current Region
13. End(XlDown)
14. Offset method
15. Strings
16. Integers
17. Variants
18. Worksheet
19. Range
20. Trim Function
21. Chr(10)
22. Call subs
23. .clear, .activate methods
24. IsEmpty Function

Henry Tran
Lee Chang
IS 520
Gove Allen

25. Set
26. Format as date/currency functions
27. <> = operators
28. Unload
29. MsgBox
30. CDate function

II. Living Costs - Henry

This portion relates to the living costs and will be broken down in the following sections:

a. Retrieving Travelling Cost Information

Retrieving Travelling Cost Information from the Internet.

To aggregate the estimated travel cost from the internet, I created three subroutines and one function to complete such task, namely *GetDataFromBudgetYourTrip(), GetCurrencySymbol(), ConversionRate()* and *FormatTemplate()*. All the subsequent subs and function were integrated into *GetDataFromBudgetYourTrip()* sub to produce the final product. At the end, the program will break the estimated costs down into different categories and present them in both the local currency of the traveling destination and the U.S. dollars.

1. Function ConversionRate(LocalCurrencyCode)

To plan well for a trip, trip planners need to know what it costs not only in terms of the U.S. dollars but also in terms of the local currency. Knowing this information may facilitate travelers to negotiate or bargain with local foreigners. This subroutine simply takes a predetermined *LocalCurrencyCode* value (i.e., CAD for Canada, USD for U.S. Dollars) and look it up on the online exchange rate tracker- xe.com and then return the conversion rate of the local currency equivalent of one U.S. Dollars.

2. Subroutine GetCurrencySymbol(LocalCurrencyCode)

This sub facilitates the cell-formatting process on the end result. Simply, to enhance readability, this sub will add the appropriate foreign currency symbol to the numbers such as £10, €20. To achieve this goal, the subroutine must complete two steps.

1) Load all the currency symbols into an array (*SymbolTable*) from the internet, specifically this webpage, http://www.xe.com/symbols.php.

2) Match the predetermined *LocalCurrencyCode* value with the local currency code in the *SymbolTable* array and return the currency symbol by assigning that character to a variable named *LocalCurrencySymbol*.

To load all the currency symbols into an array, I must first create the array. To do so, I run the *Do while … Loop* statement twice to measure the dimension of the HTML table. Since there are two separate tables

in the HTML codes, I first measure the total number of table rows (by counting the appearances of <tr> tag) and record this value in *counter1*. Then, I measure the total number of rows of the second table by the same method and record this value in *counter2*. The difference between *counter1* and *counter2* is the number of rows I need to declare for the *SymbolTable* array.

After declaring the dimensions of the *SymbolTable*, I name the cells of the first row "Currency Code","Arial Unicode MS","Unicode Decimal" in that order. The headers are mainly for keeping track of what each column is. I technically don't need to keep track of unicode decimals. However, I still do so because I think it might help. In case the currency symbol downloaded from the Internet doesn't turn out to be what I expected it to be, by converting the unicode decimals back to the symbols, I will have the same result to that effect.

The next task at hand is to scrape the wanted data from HTML codes. To complete this task, I rely on the built-in agent class module provided by Dr. Allen for the class. By running a *For ...Next* loop, the program is able copy the currency code, currency symbol and unicode decimals then place them in the appropriate columns and cells in the *SymbolTable* array.

When the *SymbolTable* array is completed, I run another loop to match the predetermined *LocalCurrencyCode* against the *SymbolTable*. Once a match is found, the subroutine will return the symbol of that local currency and place this string in the variable called *LocalCurrencySymbol*.

3. Subroutine FormatTemplate (Range as Object)

The purpose of this sub is to format a selected range of cells according to a predetermined style. In this case, it will bold and italicize the font while changing the font color to red and the cell's interior color to green.

4. Subroutine GetDataFromBudgetYourTrip()

This is the main subroutine that will retrieve the cost information from the internet as well as present such information in an organized format in a new worksheet. This process will be broken into steps as follow:

Step 1: Subroutine *GetDataFromBudgetYourTrip* quietly accesses to *BugetYourTrip.com* under a customized URL and download all the HTML codes and store them in an agent named "a".

Step 2: Retrieve the local currency code and store it in the variable named *LocalCurrencyCode.*

Step 3: Build an array (*PriceTable*) to store all the cost information with their respective category. To do so, I need to run a *Do While… Loop* to count the total number of <td> tags. Since this is the only table in the HTML codes, I don't need to run two loops as I have done in subroutine *GetCurrencySymbol()*.

Step 4: When the dimension of the array is determined, I start to fill the array with information. After having observed that there is one exception to the code of "Daily Cost" entry in the HTML code, I have decided to treat this code separately before I run any loop. Relying on

the built-in function of agent, I extract this data, place them in the first and second column of the second row, and rename "Daily Cost" to 'Daily Cost on Average per Person."

Step 5: Once irregularities are properly treated, I run a *Do While...Loop* to place appropriate data into the array as planned.

Step 6: Because the retrieved cost information is reported in foreign currency, the subroutine will then call *ConversionRate* to get the most up-to-date conversion rate and place the value in a variable named "Xrate".

Step 7: When the exchange rate is ready, a loop statement (*For… Next*) will run through the array and convert the foreign currency into U.S. Dollars using a prescribed formula and place the values in the adjacent cells. For this formula to work smoothly, it is important remove all the commas (",") that appear in the number string and the convert these strings into value by using *Val* function.

Step 8: At this point, all the information is ready for presentation. The subroutine will generate a new worksheet and name it after the name of the destination. To make it easy to find, the new worksheet will be added to the end of all current worksheets. Then, the creation of a table is underway. This subroutine will write the three column headers of the table. Namely, Category, Cost in *LocalCurrencyCode*, and Cost in USD. To further emphasize the headers, I have created a code to change the font size of the headers to 15 and call the subroutine *FormatTemplate* to apply the predetermined style.

Step 9: This subroutine will print all the data from the *PriceTable* array onto this new worksheet using a loop statement (*For … Next*) and will auto-fit all the data columns. In addition, because the daily average cost is the main indicator of cost in the table, that row of the table will be bolded as the result.

Step 10: To enhance readability, the subroutine will then format the cost number in "Cost in USD" column by adding "$" in front of the number, "," between every three digits, and two decimals at the end.

Step 11: To prevent name error if users accidentally run the code twice with the same selection of location. The subroutine at this point will rename the activesheet by adding the number of sheet count in front of the destination name.

Step 12: Lastly, I need to format the foreign currency in the same fashion as the cost information in U.S. Dollars. To achieve such effect, the subroutine will first run *LocalCurrencySymbol* to get the currency symbol and add that symbol to front of the cost number. Because Microsoft Windows does not support all currency symbols by default, this process is in fact prone to errors. To avoid the potential errors, I added an *On Error* statement which allows the code to skip formatting these cost numbers if necessary. Otherwise, the code will format in the same fashion as the "Cost in USD" column.

Henry Tran
Lee Chang
IS 520
Gove Allen

Conclusion - Lee

Through Henry's and my combined efforts, we have created a program that simplifies the research process of travel. My portion of the project allows the user the compare airfare and routes efficiently and quickly. Henry's section condenses large quantities of related data to a concise summary from the internet, providing vital information to forecast the cost of staying in a foreign country. Both sections can be easily scaled up and expanded to cover more destinations. Over the 40-50 combined hours of work for this project, both Henry and I have become extensively familiar with the web-scraping process, loops, user forms, and arrays. It is a project we plan on expanding further on our own time. We hope that all who use it can appreciate its utility as well.

Assistance

The only resources consulted were Stack Overflow and Professor Allen. Professor Allen helped me (Lee) understand how to use a split array more effectively in the project, which played a crucial part in the coding.