Jaewoo An

Professor Choi

CS101

March 20, 2012

Homework 2

The following seven tables show seven functions I have defined for this assignment.

|  |  |
| --- | --- |
| month\_to\_num\_of\_days(tmp\_year, tmp\_month) | |
| **Role** | Returns the number of days in the month (tmp\_month) of the year (tmp\_year) |
| **Explanation on how I implemented** | I used if statement to give the variable num\_of\_days (which is initially 0) different values. For February of a leap year, num\_of\_days = 29. For February of a normal year, num\_of\_days = 28. For January, March, May, July, August, October, and December, num\_of\_days becomes 31, and for all other cases, num\_of\_days = 30. |

|  |  |
| --- | --- |
| is\_valid\_date(tmp\_year, tmp\_month, tmp\_day) | |
| **Role** | Returns True if a date is valid and False if a date is invalid. |
| **Explanation on how I implemented** | Initially, variable check\_result is False. I used if statement to filter all the possible invalid dates. First, if statement checks whether the year is earlier than 1901 or later than 2100, whether the month is between 1 and 12 inclusive, and whether the day is between 1 and month\_to\_num\_of\_days (which is the last day of the corresponding month). If the date is not filtered through, then it satisfies all the requirements for validity, and thus this function returns True. |

|  |  |
| --- | --- |
| what\_day(n) | |
| **Role** | Returns whether n days from 1901.01.01 is Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday. |
| **Explanation on how I implemented** | Since each day is repeated every seven days, I used mod (%). If n%7 is 1, it means the date is exactly 7x days since 1901.01.01, and thus its day has to be the same as 1901.01.01, which is Tuesday. If n%7 = 2, then it means that the date is exactly 7x+1 days after 1901.01.01. Therefore, its day has to be a day after Tuesday, which is Wednesday. In the same way, it is obvious that it is Thursday if n%7 = 3, Friday if n%7 = 4, Saturday if n%7 = 5, Sunday if n%7 = 6, and Monday if n%7 = 0. |

|  |  |
| --- | --- |
| date\_to\_num(tmp\_year, tmp\_month, tmp\_day) | |
| **Role** | Returns how many days have passed since 1901.01.01. |
| **Explanation on how I implemented** | I mixed if statement and for loop to calculate the number of days since the year 1901 to the year (tmp\_year). For each year, if it is a leap year, I added 366 days to the variable num\_of\_days (initially 0) and 365 days otherwise. The loop stops when it reaches the year tmp\_year, having counted num\_of\_days from 1901.01.01 to (tmp\_year-1).12.31 (since it has not counted 1901.01.01 itself yet). For each month up to tmp\_month, I used month\_to\_num\_of\_days function to find out how many days are in the corresponding month and added the result to the variable num\_of\_days. So far, the function has counted num\_of\_days from 1901.01.01 to tmp\_year.(tmp\_month-1).31. Finally, I added tmp\_day to num\_of\_days, making the function to fully count the number of days since 1901.01.01 to tmp\_year.tmp\_month.tmp\_day. |

|  |  |
| --- | --- |
| num\_to\_date(n) | |
| **Role** | Returns the date n days since 1901.01.01. |
| **Explanation on how I implemented** | First, I added 1901 to tmp\_year and 1 to tmp\_month. While n > 365 (meaning a year), if the corresponding year is a leap year and n > 366, I added 1 to tmp\_year and subtractd 366 from n, and if the corresponding year is not a leap year, I added 1 to tmp\_year and subtracted 365 from n. This loop is continued until n <= 365, meaning the tmp\_year is determined. I used for loop to subtract the number of days in each month from n and add 1 to tmp\_month until n < the number of days in the corresponding month. Finally, I added n to tmp\_day. |

|  |  |
| --- | --- |
| date\_after\_days(base\_year, base\_month, base\_day, number\_of\_days) | |
| **Role** | Returns the date a certain amount of days (number\_of\_days) since the base date (base\_year.base\_month.base\_day). |
| **Explanation on how I implemented** | I first used date\_to\_num function to find out the number of days passed since 1901.01.01 to the base date, and I added the result to the variable number\_of\_days, and subtracted 1 because 1901.01.01 was counted twice (it was counted already in number\_of\_days). Using the function num\_to\_date (number\_of\_days), I could easily get the date. |

|  |  |
| --- | --- |
| num\_of\_days\_between\_dates( base\_year, base\_month, base\_day, new\_year, new\_month, new\_day ) | |
| **Role** | Returns how many days have passed since the base date to the new date. |
| **Explanation on how I implemented** | I set num\_of\_days = 1 because I have to include the base date also when I count.  I found out the number of days since 1901.01.01 to the new date using date\_to\_num function, and added the result to num\_of\_days. Using date\_to\_num function again, I found out the number of days since 1901.01.01 to the base date, and subtracted the result from num\_of\_days. () |

**Sample Execution Results**









