Alan Padilla Chua
axp141330
CS 3340.002

Final Project Report

Problem:

For the Final Project it is required of us to basically combine all other previous programs to form a super program that is able to take in a year inputted by the user. With this year our program must output every full moon that occurs in that year but also output weather a full moon is a blue moon monthly and or if a seasonal blue moon occurs as well. The most complicated part of this program is the need to calculate seasonal moons. This uses the leap year function to determine if a year is leap in accounting for the extra julian date. Also the julian date conversion program to output the Day,Month and Year of that julian date and finally the full moon program that outputs every full moon of a requested year. I choose my reference year of 1970 so my program only performs calculations on years equal or greater than 1970.

Approach:

Since my fullmoon program built the framework in order to calculate full moons and implementing my julian date program with the leap year. I was able to keep that framework basically intact. The main change that I did make was with my season calculator program. Instead of outputting the individual date and time for each season, I kept that information hidden to the user and just stored the julian date of each season into an array. These julian date are obtained in the beginning of the program since seasoncal is the first function that is called.

SeasonCal function first finds the amount of years in between the reference year and the requested year. This is used to run a loop from the reference year to the requested year. The loop sends each year into the leapyear function and increments a counter if the year is a loop year. Once this number is found then it's multiplied by 366(numberofleapyears * 366) this is added with the number of non leap years multiplied by 365 (Non-leap years *365). So this gives us actual days = (#ofleapyears * 366) + (#ofNon-leap years *365). Then the number of years in between is multiplied by 365 (delta * 365). Finally (delta*365) is subtracted from Actual days ((#ofleapyears * 366) + (#ofNon-leap years *365). This value is added to the reference julian date that is loaded from memory. This julian date is then sent to the functions I defined in program 4. This process is done for each reference julian date. The hour is calculated with the reference number of the season added by delta multiplied by 5 (delta*5). The min is calculated the same way by loading the reference number for min and adding by delta multiplied by 50 (delta*50). Once these numbers are added and corrected for overflow and converted to 12 hour clock time then all values are outputted for each individual season.

With seasoncal calculating the julian dates for each season completed it then goes into the FullMooncal function. In order to calculate the monthly and seasonal bluemoons, I decided that it was best to split the functionality into two different functions. One calculated the monthly blue moons and the other calculated seasonal blue moons.

The easiest one to calculate is the monthly bluemoons. For this function I load the previouses moon index value and then with the julian date of the current fullmoon I calculate the month of which the julian date is in. If both those months are the same then it returns a 1 if not then it returns a 0. The pseudocode looks like this:

```
    boolean BlueMonthCal(int prevMonthIndex, int curJulianDate){
    if(leapyear(){
        for(int i = 0; i < leapYearArray.length; i++){
            if(curJulianDate< leapYeaArray[i]){
                if((leapYearArray[i]-1) == prevMonthIndex){
                    return true;
                }
                else
                    return false;}
        }
        else
            for(int i = 0; i < YearArray.length; i++){
            if(curJulianDate< lYeaArray[i]){
                if((YearArray[i] -1_== prevMonthIndex){
                    return true;
                }
                else
                    return false;
        }
}
```

For the BlueSeasonalMoon function it required a bit more calculations and registers to keep track of. First before the function is called inside my fullmoon function I set 4 counters set to zero. One counter for each season. Then during the calculation of fullmoon I send the julian date into my BlueSeasonalMoon function after going into the bluemoonth function. Once inside the blue seasonal function. It is here where i compare the julian date of the full moon to find the correct season that it belongs to. If it's greater than or equal to the start of spring and less than the start of summer then it must be in spring. This same logic is continued for the rest of the season. So if the julian date is found to be in a season the counter for that specific season is increased. Once the counter reaches 3 then that julian date of the full moon is stored into an array because we will need it in order to identify the full moon that is considered the blue seasonal moon. Once the counter reaches 4 it is then that we jump into another if statement that goes ahead and calls the output function to print to the user that that full moon is the seasonal blue moon of that corresponding season. The pseudo code looks as follows:

```
void BlueSeasonCal(){
if(JD >=79 && JD < 172){
    springMoon++;
    if(springMoon ==3){
    JD = BMSP;}
    if(springMoon ==4){
    Output: Seasonal Blue Moon:
    }
}
if(JD >= 172 && JD < 266){
    summerMoon++;
    if(summerMoon == 3){
    JD = BMSU;}
    if(summerMoon ==4){
    Output: Seasonal Blue Moon:
    }
}
if(JD >= 226 && JD < 356){
    fallMoon++;
    if(fallMoon == 3){
    JD = BMFL;}
    if(fallMoon ==4){
    Output: Seasonal Blue Moon:
    }
}
if(JD >= 356 || JD < 79){
    winterMoon++;
    if(winterMoon == 3){
    JD = BMWI;}
    if(winterMoon ==4){
    Output: Seasonal Blue Moon:
    }
}
```

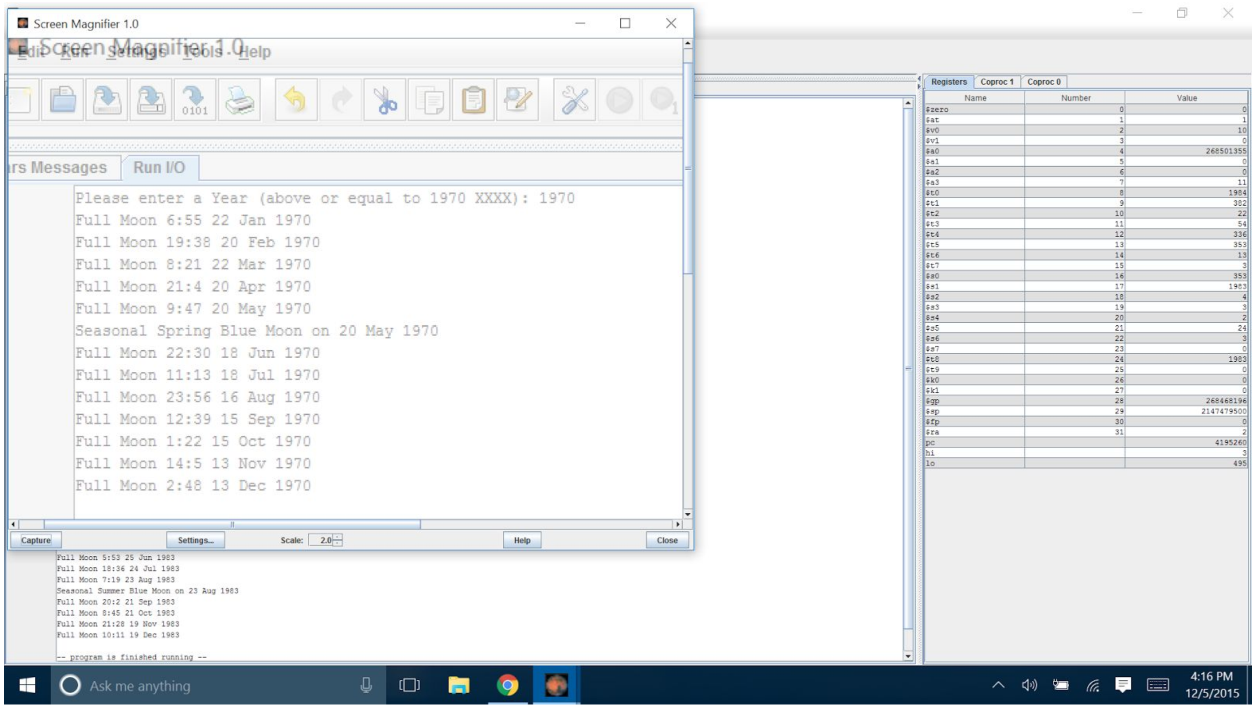
Finally the process is complete. These functions are called within the FullMoonCalculator function. The function is based on loops. To find the next full moon we increment 29 days, 12 hours and 43 minutes and adjust for overflow. This is done until the reference year is equal to the requested year. The leap year function is also used in determining the correct amount of days to subtract depending if that year is a leap year or not. Once The RefYear is equal to requested Year the algorithm for output is essentially the same in incrementing to output all full moons until the year is passed. However the dates and times calculated to accumulate an error offset the further the requested year is from the reference year. The finale pseudo code is as follows including the function calls for monthly and seasonal full moons and output.

pseudo code:

```
springMoon=0;
summerMoon=0;
fallMoon=0;
winterMoon=0;
while(RefYear < ReqYear){
    RefDay += 29;
    RefHour += 12;
    RefMin +=43;
```

```
        if(RefMin >= 60){
                RefMin -= 60;
                RefHour++;}
        if(RefHour >= 24){
                RefHour -=24;
                RefDay++;}
        if(RefYear is leapyear && RefDay > 366){
                RefYear++;
                RefDay -= 366;}
        else if (RefDay > 365){
                RefYear++;
                RefDay-=365;}
        MonthBlueMoon(RefDay);
        SeasonalBlueMoon(RefDay);
        Output();
}
```

Evidence:



```
Please enter a Year (above or equal to 1970 XXXX): 1970
Full Moon 6:55 22 Jan 1970
Full Moon 19:38 20 Feb 1970
Full Moon 8:21 22 Mar 1970
Full Moon 21:4 20 Apr 1970
Full Moon 9:47 20 May 1970
Seasonal Spring Blue Moon on 20 May 1970
Full Moon 22:30 18 Jun 1970
Full Moon 11:13 18 Jul 1970
Full Moon 23:56 16 Aug 1970
Full Moon 12:39 15 Sep 1970
Full Moon 1:22 15 Oct 1970
Full Moon 14:5 13 Nov 1970
Full Moon 2:48 13 Dec 1970
```

```
Full Moon 5:53 25 Jun 1983
Full Moon 18:36 24 Jul 1983
Full Moon 7:19 23 Aug 1983
Seasonal Summer Blue Moon on 23 Aug 1983
Full Moon 20:2 21 Sep 1983
Full Moon 8:45 21 Oct 1983
Full Moon 21:28 19 Nov 1983
Full Moon 10:11 19 Dec 1983

-- program is finished running --
```

Name	Number	Value
$zero	0	0
$at	1	1
$v0	2	10
$v1	3	0
$a0	4	268501355
$a1	5	0
$a2	6	0
$a3	7	11
$t0	8	1984
$t1	9	382
$t2	10	22
$t3	11	54
$t4	12	336
$t5	13	353
$t6	14	13
$t7	15	3
$s0	16	353
$s1	17	1983
$s2	18	4
$s3	19	3
$s4	20	2
$s5	21	24
$s6	22	3
$s7	23	0
$t8	24	1983
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468196
$sp	29	2147479500
$fp	30	0
$ra	31	2
pc		4195260
hi		3
lo		495

Screen Magnifier 1.0
Edit Run Settings Tools Help
Messages Run I/O
Capture Settings... Scale: 2.0 Help Close
Registers Coproc 1 Coproc 0
Ask me anything
4:16 PM
12/5/2015

Input year: 1970

OutPut:

Please enter a Year (above or equal to 1970 XXXX): 1970

Full Moon 6:55 22 Jan 1970

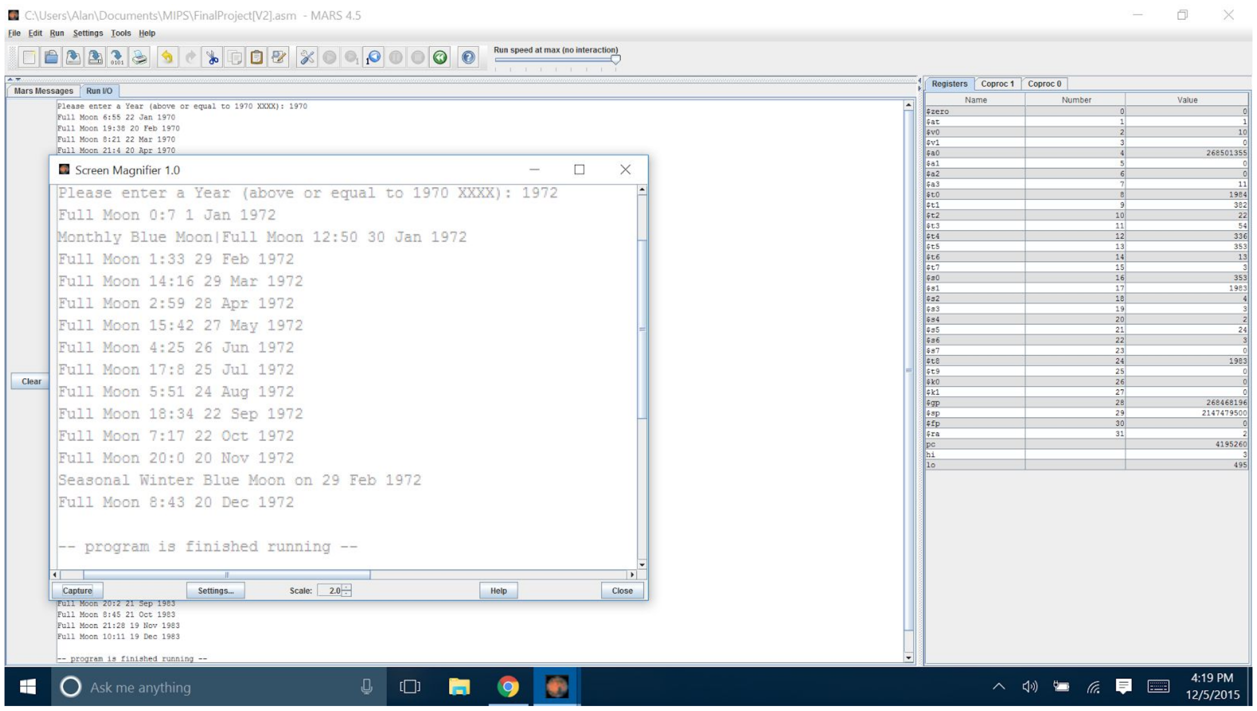
Full Moon 19:38 20 Feb 1970

Full Moon 8:21 22 Mar 1970

Full Moon 21:4 20 Apr 1970

Full Moon 9:47 20 May 1970
Seasonal Spring Blue Moon on 20 May 1970
Full Moon 22:30 18 Jun 1970
Full Moon 11:13 18 Jul 1970
Full Moon 23:56 16 Aug 1970
Full Moon 12:39 15 Sep 1970
Full Moon 1:22 15 Oct 1970
Full Moon 14:5 13 Nov 1970
Full Moon 2:48 13 Dec 1970

Input year: 1972



```
Please enter a Year (above or equal to 1970 XXXX): 1970
Full Moon 6:55 22 Jan 1970
Full Moon 19:38 20 Feb 1970
Full Moon 8:21 22 Mar 1970
Full Moon 21:4 20 Apr 1970
```

```
Please enter a Year (above or equal to 1970 XXXX): 1972
Full Moon 0:7 1 Jan 1972
Monthly Blue Moon|Full Moon 12:50 30 Jan 1972
Full Moon 1:33 29 Feb 1972
Full Moon 14:16 29 Mar 1972
Full Moon 2:59 28 Apr 1972
Full Moon 15:42 27 May 1972
Full Moon 4:25 26 Jun 1972
Full Moon 17:8 25 Jul 1972
Full Moon 5:51 24 Aug 1972
Full Moon 18:34 22 Sep 1972
Full Moon 7:17 22 Oct 1972
Full Moon 20:0 20 Nov 1972
Seasonal Winter Blue Moon on 29 Feb 1972
Full Moon 8:43 20 Dec 1972

-- program is finished running --
```

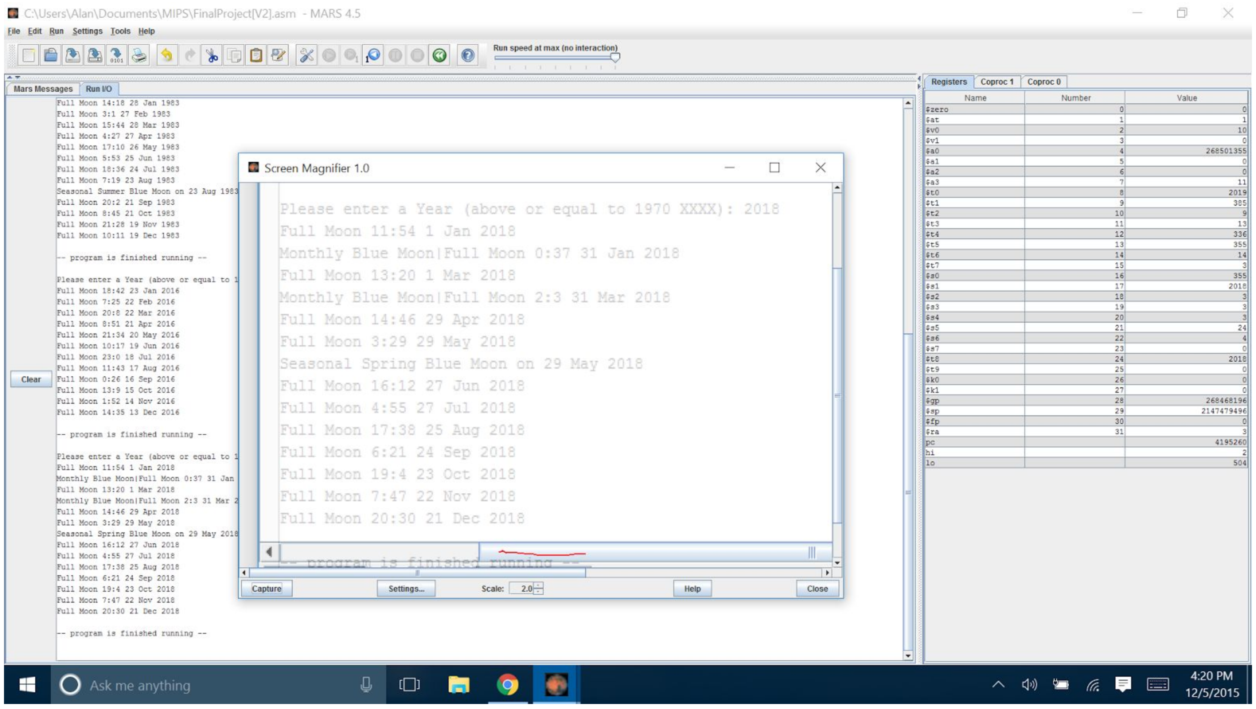
```
Full Moon 20:2 21 Sep 1983
Full Moon 8:45 21 Oct 1983
Full Moon 21:28 19 Nov 1983
Full Moon 10:11 19 Dec 1983

-- program is finished running --
```

Name	Number	Value
$zero	0	0
$at	1	1
$v0	2	10
$v1	3	0
$a0	4	268501355
$a1	5	0
$a2	6	0
$a3	7	11
$t0	8	1984
$t1	9	382
$t2	10	22
$t3	11	54
$t4	12	336
$t5	13	353
$t6	14	13
$t7	15	3
$s0	16	353
$s1	17	1983
$s2	18	4
$s3	19	3
$s4	20	2
$s5	21	24
$s6	22	3
$s7	23	0
$t8	24	1983
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468196
$sp	29	2147479500
$fp	30	0
$ra	31	2
pc		4195260
hi		3
lo		495

Output:
Please enter a Year (above or equal to 1970 XXXX): 1972
Full Moon 0:7 1 Jan 1972
Monthly Blue Moon|Full Moon 12:50 30 Jan 1972
Full Moon 1:33 29 Feb 1972
Full Moon 14:16 29 Mar 1972
Full Moon 2:59 28 Apr 1972
Full Moon 15:42 27 May 1972
Full Moon 4:25 26 Jun 1972
Full Moon 17:8 25 Jul 1972
Full Moon 5:51 24 Aug 1972
Full Moon 18:34 22 Sep 1972

Full Moon 7:17 22 Oct 1972
Full Moon 20:0 20 Nov 1972
Seasonal Winter Blue Moon on 29 Feb 1972
Full Moon 8:43 20 Dec 1972



```
Full Moon 14:18 28 Jan 1983
Full Moon 3:1 27 Feb 1983
Full Moon 15:44 28 Mar 1983
Full Moon 4:27 27 Apr 1983
Full Moon 17:10 26 May 1983
Full Moon 5:53 25 Jun 1983
Full Moon 18:36 24 Jul 1983
Full Moon 7:19 23 Aug 1983
Seasonal Summer Blue Moon on 23 Aug 1983
Full Moon 20:2 21 Sep 1983
Full Moon 8:45 21 Oct 1983
Full Moon 21:28 19 Nov 1983
Full Moon 10:11 19 Dec 1983

-- program is finished running --

Please enter a Year (above or equal to 1
Full Moon 18:42 23 Jan 2016
Full Moon 7:25 22 Feb 2016
Full Moon 20:8 22 Mar 2016
Full Moon 8:51 21 Apr 2016
Full Moon 21:34 20 May 2016
Full Moon 10:17 19 Jun 2016
Full Moon 23:0 18 Jul 2016
Full Moon 11:43 17 Aug 2016
Full Moon 0:26 16 Sep 2016
Full Moon 13:9 15 Oct 2016
Full Moon 1:52 14 Nov 2016
Full Moon 14:35 13 Dec 2016

-- program is finished running --

Please enter a Year (above or equal to 1
Full Moon 11:54 1 Jan 2018
Monthly Blue Moon|Full Moon 0:37 31 Jan
Full Moon 13:20 1 Mar 2018
Monthly Blue Moon|Full Moon 2:3 31 Mar 2
Full Moon 14:46 29 Apr 2018
Full Moon 3:29 29 May 2018
Seasonal Spring Blue Moon on 29 May 2018
Full Moon 16:12 27 Jun 2018
Full Moon 4:55 27 Jul 2018
Full Moon 17:38 25 Aug 2018
Full Moon 6:21 24 Sep 2018
Full Moon 19:4 23 Oct 2018
Full Moon 7:47 22 Nov 2018
Full Moon 20:30 21 Dec 2018

-- program is finished running --
```

```
Please enter a Year (above or equal to 1970 XXXX): 2018
Full Moon 11:54 1 Jan 2018
Monthly Blue Moon|Full Moon 0:37 31 Jan 2018
Full Moon 13:20 1 Mar 2018
Monthly Blue Moon|Full Moon 2:3 31 Mar 2018
Full Moon 14:46 29 Apr 2018
Full Moon 3:29 29 May 2018
Seasonal Spring Blue Moon on 29 May 2018
Full Moon 16:12 27 Jun 2018
Full Moon 4:55 27 Jul 2018
Full Moon 17:38 25 Aug 2018
Full Moon 6:21 24 Sep 2018
Full Moon 19:4 23 Oct 2018
Full Moon 7:47 22 Nov 2018
Full Moon 20:30 21 Dec 2018

-- program is finished running --
```

Name	Number	Value
$zero	0	0
$at	1	1
$v0	2	10
$v1	3	0
$a0	4	268501355
$a1	5	0
$a2	6	0
$a3	7	11
$t0	8	2019
$t1	9	385
$t2	10	9
$t3	11	13
$t4	12	336
$t5	13	355
$t6	14	14
$t7	15	3
$s0	16	355
$s1	17	2018
$s2	18	3
$s3	19	3
$s4	20	3
$s5	21	24
$s6	22	4
$s7	23	0
$t8	24	2018
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468196
$sp	29	2147479496
$fp	30	0
$ra	31	3
pc		4195260
hi		2
lo		504

Input Year: 2018
OutPut:
Please enter a Year (above or equal to 1970 XXXX): 2018
Full Moon 11:54 1 Jan 2018
Monthly Blue Moon|Full Moon 0:37 31 Jan 2018
Full Moon 13:20 1 Mar 2018
Monthly Blue Moon|Full Moon 2:3 31 Mar 2018
Full Moon 14:46 29 Apr 2018
Full Moon 3:29 29 May 2018
Seasonal Spring Blue Moon on 29 May 2018
Full Moon 16:12 27 Jun 2018
Full Moon 4:55 27 Jul 2018
Full Moon 17:38 25 Aug 2018
Full Moon 6:21 24 Sep 2018
Full Moon 19:4 23 Oct 2018
Full Moon 7:47 22 Nov 2018
Full Moon 20:30 21 Dec 2018