```
Unit Misc(2);
{$U-}
Interface

uses MemTypes;

Function DayofWeek(JulianDay,Year:Integer):DateStr;
Function DeltaTime(JD1,JD2,Year1,Year2:Integer;Time1,Time2:LongInt):LongInt;
Function SNumToChar(Num:Integer):Char;
Function SCharToNum(Letter:Char):Integer;
Function SecFromMid(Time:String[7]):LongInt;
Procedure JulianDate(Date:String[10]; VAR JulianDay,Year:Integer);
Procedure WaitaBit((Delay:Integer));

Implementation

Function DayofWeek((JulianDay,Year:Integer):DateStr);
    {Given the Julian day and the year this will give you the day of the week for the date
       given.  This is good for years 1952 to 2023}
    {Rule for leap years: divisible by 4, except 00 unless divisible by 400}
VAR
    DoW         : Integer
    YearStart   : Array [1952..2020] of Integer;
Begin
    {Set up the Year Start Array}
    YearStart[1952] := 3; YearStart[1953] := 5; YearStart[1954] := 6; YearStart[1955] := 7;
    YearStart[1956] := 1; YearStart[1957] := 3; YearStart[1958] := 4; YearStart[1959] := 5;
    YearStart[1960] := 6; YearStart[1961] := 1; YearStart[1962] := 2; YearStart[1963] := 3;
    YearStart[1964] := 4; YearStart[1965] := 6; YearStart[1966] := 7; YearStart[1967] := 2;
    YearStart[1968] := 2; YearStart[1969] := 4; YearStart[1970] := 5; YearStart[1971] := 6;
    YearStart[1972] := 7; YearStart[1973] := 2; YearStart[1974] := 3; YearStart[1975] := 4;
    YearStart[1976] := 5; YearStart[1977] := 7; YearStart[1978] := 1; YearStart[1979] := 2;
    YearStart[1980] := 3; YearStart[1981] := 5; YearStart[1982] := 6; YearStart[1983] := 7;
    YearStart[1984] := 1; YearStart[1985] := 3; YearStart[1986] := 4; YearStart[1987] := 5;
    YearStart[1988] := 6; YearStart[1989] := 1; YearStart[1990] := 2; YearStart[1991] := 3;
    YearStart[1992] := 4; YearStart[1993] := 6; YearStart[1994] := 7; YearStart[1995] := 1;
    YearStart[1996] := 2; YearStart[1997] := 4; YearStart[1998] := 5; YearStart[1999] := 6;
    YearStart[2000] := 7; YearStart[2001] := 2; YearStart[2002] := 3; YearStart[2003] := 4;
    YearStart[2004] := 5; YearStart[2005] := 7; YearStart[2006] := 1; YearStart[2007] := 2;
    YearStart[2008] := 3; YearStart[2009] := 5; YearStart[2010] := 6; YearStart[2011] := 7;
    YearStart[2012] := 1; YearStart[2013] := 3; YearStart[2014] := 4; YearStart[2015] := 5;
    YearStart[2016] := 6; YearStart[2017] := 1; YearStart[2018] := 2; YearStart[2019] := 3;
    YearStart[2020] := 4; YearStart[2021] := 6; YearStart[2022] := 7; YearStart[2023] := 1;
    {There must be a simpler way but I haven't thought of it yet}
    {Now for the heart of the matter}
    DoW := ((YearStart[Year] + JulianDay - 1) mod 7) + 1;
    Case DoW of
        1: DayofWeek := 'Sunday    ';
        2: DayofWeek := "Monday    ';
        3: DayofWeek := "Tuesday';
        4: DayofWeek := 'Wednesday';
        5: DayofWeek := 'Thursday';
        6: DayofWeek := 'Friday';
        7: DayofWeek := 'Saturday';
    end;
End;

Function DeltaTime((JD1,JD2,Year1,Year2:Integer;Time1,Time2:LongInt):LongInt);
    {Note: This function is only good for +/- 68 years due to LongInt size}
VAR JDf,JDi,Yeayf,Yeari,DelDay  :Integer;
    DelSec                      : LongInt;
Begin
    JDi := JD1; JDf := JD2; Yeari := Year1; Yearf := Year2;
    While Yeari<>Yearf DO
        IF Yearf>Yeari THEN Begin
                                JDf := JDf + 365;
                                Yearf := Yearf-1;
                            end;
```

```
                        ELSE Begin
                                JDi := JDi + 365;
                                Yeari := Yeari - 1;
                             end;
    DelDay := JDf - JDi;
    DelSec := Time2 - Time1;
    DeltaTime := (DelDay*24*3600) + DelSec;
End;

Function SNumToChar{(Num:Integer):Char};
Begin
    NumToChar := Chr(Num+48);
End;

Function SCharToNum{(Letter:Char):Integer};
Begin
    CharToNum := ORD(Letter-48);
End;

Function SecFromMid{(Time:String[7]):Longint};
Begin
    SecFromMid := CharToNum(Time[7]);
    SecFromMid := SecFromMid + CharToNum(Time[6])*10;
    SecFromMid := SecFromMid + CharToNum(Time[4])*60;
    SecFromMid := SecFromMid + CharToNum(Time[3])*600;
    SecFromMid := SecFromMid + CharToNum(Time[2])*3600;
    SecFromMid := SecFromMid + CharToNum(Time[1])*36000;
End;

Procedure JulianDate{(Date:String[10]; VAR JulianDay,Year:Integer)};
VAR
    Day,Month,Index : Integer
    DaysInMonth     : Array [1..12] of Integer;
Begin
    {A little setup initially}
    DaysInMonth[1] := 31;DaysInMonth[2] := 28;DaysInMonth[3] := 31;DaysInMonth[4] := 30;
    DaysInMonth[5] := 31;DaysInMonth[6] := 30;DaysInMonth[7] := 31;DaysInMonth[8] := 31;
    DaysInMonth[9] := 30;DaysInMonth[10] := 31;DaysInMonth[11] := 30;DaysInMonth[12] := 31;
    {Compute Year First}
    Year := CharToNum(Date[10]);
    Year := Year + CharToNum(Date[9])*10;
    Year := Year + CharToNum(Date[8])*100;
    Year := Year + CharToNum(Date[7])*1000;
    {Now for Day, Then Month}
    Day := (CharToNum(Date[4])*10) + CharToNum(Date[5]);
    Month := (CharToNum(Date[1])*10) + CharToNum(Date[2]);
    {Leap Year Check}
    IF Year Mod 4 = 0 THEN DaysInMonth[2] := DaysInMonth[2] + 1;
    {Now Compute Julian Day}
    JulianDay := 0;
    Index := 1;
    While Index<Month Do Begin
        JulianDay := JulianDay + DaysInMonth[Index]; Index := Index + 1; end;
    JulianDay := JulianDay + Day;
End;

Procedure WaitaBit{(Delay:Integer)};
VAR
    Start,Finis,Now : ?
Begin
    {get time}
    Finis := Start + Delay;
    Now := {get time}
    While Now<Finis Do Begin
        Now := Get time};
        SystemTask;
        end;
End;
```