27 June 2017 -- Computer Architectures -- part 1/2

Matr, Last Name, First Name

Traveling all around the world as well as the need to setting up meetings involving participants from different countries has soon or later let a question to be asked: "what time is it now in…?". It is well known that the origin of the different time zones has been conventionally set in Greenwich, UK, which is also known as GMT+0. It is also well known that besides 24 main time zones, each one spaced by the other by one hour, there are single regions in some countries (like Australia), which do have additional sub-time-zones, spaced by fractions of hour. In this text, for sake of simplicity, we will not consider these sub-time-zones, but only the 24 main time zones, which are numbered -12 … +12. In fact, it is well known that, conventionally, the time zone corresponding to the "change of the day line" (CDL), is positioned in correspondence of GMT+ (or -) 12; so, for example, if at GMT+0 it is 2 am of day xx, then at GMT+11 it will be 1 pm of the same day xx, at GMT -11 it will be 3 pm of day xx-1. The CDL is a singularity point, as while it is a fact that the hour (as per the previous example) will be 2 pm, then, the day of the year is determined by the direction someone is moving from/to. More specifically, if moving from GMT+11 towards CDL, then the 12 hours will be subtracted, thus becoming 2 pm of day xx-1; on the other hand, if moving from GMT-11 to CDL, then the 12 hours will be added thus becoming 2 pm of the same day xx. For sake of simplicity in this text we will consider that we do not have to manage hours/queries about locations placed in correspondence of CDL.

Another issue to be considered, when calculating the hour at a different place, is the existence (as well as starting/ending) of the "daylight saving time" (DST) period. For example, in Central Europe daylight saving time starts the last weekend of March and ends the last weekend of October. Daylight saving time period is not only determined by the time zone (plus some local political choices), but also from the hemisphere where a country is located. For sake of simplicity (in the real world the picture is slightly more complex), let us assume that the world is divided in 3 main zones: North, Central and South Hemispheres, where only the North and South do have saving times. In addition, let us assume that the saving time switching (back or forth) day and hour is synchronous for all the countries belonging to the same hemisphere, and is referred to the day and hour of the GMT+0 time zone without considering any "correction" due to DST.

Given the above, let us assume to have already available in one array DAYLIGHT DW 4 DUP(?), the information about the switch back and forth to/from DST of the North and South Hemispheres of a given year in the format: HCmmmmdd dddhhhhh where ddddd are the day bits, mmmm the month, hhhhh the hour (referred to GMT+0), where H=0 stands for the north hemisphere and =1 for the south and where C=0 if it is the starting of DST and =1 if it is the end of DST for that year. For simplicity, this information is referred to GMT+0, i.e. dates and hours.

It is requested to write a program receiving in input (by assumption all input data have to be considered correct in their values):

- Date and hour (no minutes, just the hour), expressed in 24 hours format;
- A name of a city together with its time zone expressed as a signed number in the range -11 .. +11 (for sake of simplicity no cities on the CDL are considered), as well as its position N, C, S standing for North/Central/South Hemisphere.

and producing in output the date and hour at that location.

More in the detail, tasks to be implemented and corresponding points (only fully completed items will be considered to award points). Solve only one between A, B and C. Please consider that the reference year is 2017.

- Item A: Compute the date and hour at that location, without taking account the daylight saving time and hemisphere data; all dates should be correct and, for sake of simplicity it is assumed that the dates are within 2017 but not earlier than January 3rd and not later than December 27th. POINTS → 20
- Item B: Compute the date and hour at that location, taking into account the daylight saving time and hemisphere data; for sake of simplicity, the day before the given date can be expressed as date "-1 day" and the day after as date "+1" (e.g. given date = 1 october, the day before could be expressed as "1 october -1 day" and the day after as "1 october +1 day"). POINTS → 25
- Item C: Compute the date and hour at that location, taking into account the daylight saving time and hemisphere data; all dates should be correct (i.e. month and day) and, for sake of simplicity it is assumed that the dates are within 2017 but not earlier than January 3rd and not later than December 27th. POINTS → 31

INPUT:
- Local city (name, date, hour, zone, hemisphere)
- Remote city (name, zone hemisphere)

OUTPUT
- date and hour of remote city

27 June 2017 -- Computer Architectures -- part 1/2

Matr, Last Name, First Name

- **Bonus Item 1**: return the information if the current time of source and destination cities are both in a "possible time compatible window" for a business meeting. A "possible time compatible window" is defined in as from 8.00 to 18.00 (this does not check the day of the week); please do NOT check the minutes. POINTS → +3;
- **Bonus Item 2**: (ONLY IF BONUS ITEM 1 HAS BEEN SOLVED) in addition to Bonus Item 1, return the information about how long (how many hours) a meeting started immediately will remain in the "possible time compatible window"; POINTS → +3;

Examples. Assuming that the DAYLIGHT array contains the following information (dates and hours do not necessarily matching with real world):

- North hemisphere: beginning of DST = march 26 @ 2 hours (in the morning); end of DST = october 29 @ 2 hours (all times are GMT+0 and not including the DST correction)
- South hemisphere: end of DST = april 2 @ 2 hours (in the morning); beginning of DST = october 1 @ 2 hours (all times are GMT+0 and not including the DST correction)

Local city: Torino, N, GMT+1, date June 27, hour=10.00 (the program will verify that Torino is on DST)
USER INPUT -> Remote city: Sydney, S, GMT+10 → (the program will verify that Sydney is NOT on DST); June 27, hour=18.00
This hour is currently in a "time compatible window for a meeting". There are still 0 hours of permanence in the "time compatible window for a meeting".

Local city: London, N, GMT, date June 30, hour=15.00 (the program will verify that London is on DST)
Remote city: New Orleans, N, GMT-6 → (the program will verify that New Orleans is on DST); June 30, hour=9.00
This hour is currently in a "time compatible window for a meeting". There are still 3 hours of permanence in the "time compatible window for a meeting".

Local city: Istanbul, N, GMT+2, date June 30, hour=19.00 (the program will verify that Istanbul is on DST)
Remote city: Sydney, S, GMT+10 → (the program will verify that Sydney is NOT on DST); July 1, hour=2.00
This hour is NOT currently in a "time compatible window for a meeting".

HINTS (observe that)

- **Consider DST as a "temporary +1" with respect to the time zone.**
- **Bonus Item 2 is solvable either by direct computation or by a trial and error approach with multiple checks similar as these of Bonus 1.**

IMPORTANT NOTES AND REQUIREMENTS (SHARP)

- It is not required to provide the/an optimal solution, but a working and clear one using all information provided.
- It is required to write at class time a short & clear explanation of the algorithm and significant instruction comments.
- Input-output is not necessary in class-developed solution, but its implementation is mandatory for the oral exam.
- Minimum score to "pass" this part is 15 (to be averaged with second part and to yield a value at least 18)
- **To avoid misunderstandings, please consider that, as in the previous calls of the last 5 years (at least) the final score reflects the overall evaluation of the code, i.e., fatal errors, such as division by zero (etc) make it impossible to reach 30 or larger scores. Specifically, at oral exam, students will request the evaluation of some or all the parts that they have solved; prior proceeding to the correction, the points of the parts to be corrected will be added up and bounded to max 34. The final score, after the correction of students' requested items, will be "cut off" to 32.**

REQUIREMENTS ON THE I/O PART TO BE DONE AT HOME

- The databases (if any) have to be defined and initialized inside the code; in this case, all data related to object purchase have to be input from the keyboard **(i.e. NOT stored in the array)**
- All inputs and outputs should be in readable ASCII form (no binary is permitted).

Please use carbon copy ONLY (NO PICTURES ARE ALLOWED) and retain one copy for home implementation and debug. At the end of the exam please give to professors all the sheets of your solution. Missing or late sheet will not be evaluated. Please provide your classroom submitted solution with several explanatory and significant comments. Please remember that only what has been developed at class time can and will be evaluated at oral time and that it is necessary to write the instructions of the program and not just the description of the algorithm. When coming to oral discussion, please clearly mark in red on your "classroom" copy, all modifications. Please also provide an error-free and running release of the solution, as well as with its printed list of instructions. Please consider that the above are necessary but not sufficient requirements to success the exam, since the final evaluation will be based on a number of parameters. FAILURE TO ACCOMPLISH ALL THE ABOVE NECESSARY REQUIREMENTS WILL CAUSE NO-QUESTION-ASKED AND IMMEDIATE REJECTION.