FIT1008 – Intro to Computer Science
Assessed Prac 1 – Weeks 3 and 4

Semester 2, 2016

Objectives of this practical session

To be able to write MIPS programs involving lists, local variables, and functions.
Note:

- **Local variables must be stored on the runtime stack.**
- **For all the MIPS programs you should first implement a Python version, and work out a faithful translation into MIPS. Both, the Python and MIPS code are required for your prac to be marked**
- **Use only instructions on the MIPS reference sheet and use comments to document each piece of code.**
- **You may copy and paste your code between tasks where you need to reuse code.**
- **Create a new file/module for each task or subtask.**
- **Name your files task[num]_[part] to keep them organised.**

Task 1 [3 marks]

In the Gregorian calendar a *year* is a *leap year* if the *year* is divisible by 4 but not divisible by 100, or if the *year* is divisible by 400.

(a) Write a Python program `task1_a.py`, which reads in a year (i.e., an integer ≥ 1582), and if the *year* is a leap year prints "Is a leap year", otherwise prints "Is not a leap year". In your module you should include a function with the signature `def is_leap_year(year)` that returns true if it is a leap year otherwise it returns false. You should avoid doing any input or output in this function, instead do the input and output as part of the **main** function..

Run `test_task1_a.py` to ensure it works properly. Alternatively you can write your own test.

(b) Write a MIPS program which implements `task1_a.py` (you do not need to use functions).

Task 2 [4 marks]

(a) Write a Python program `task2_a.py`, which performs the steps below.

- Reads in the size of the `the_list`.
- Reads in all the items of `the_list`.
- Prints `the_list` out in reverse order.

(b) Write a MIPS program which implements `task2_a.py` faithfully – there is no need to use functions at this stage.

Task 3 [3 marks]

(a) Write a Python program `task3_a.py`, which does the following:

- Reads in the size of the `the_list`.
- Reads in all the items of `the_list`, storing them in a list.
- Prints the average value of the list

(b) Write a MIPS program which implements `task3_a.py` faithfully without the use of functions.

(c) Reimplement part (b) as a function in MIPS.

CHECKPOINT

(You should reach this point during week 3)

Background

The bureau of climate research has a device that records the average temperature in the city for each one of the days in a month. The office is interested in understanding climate variation and has commissioned you to design a collection of programs to help analyse the data collected. The programs will run in a small portable device based on a MIPS processor. For each task, first design the algorithm to be used in Python then write the algorithms in MIPS. In each task, the input list is a list containing a number for each day in the month, one temperature record per day.
An example of one such list is:
`26, 18, 22, 20, 13, 22, 19, 22, 20, 27, 18, 24, 15, 28, 26, 27, 20, 21, 23, 24, 27, 26, 15, 23, 22, 20, 23, 17, 18, 18`
This list is to be read in at the beginning of each task. The result of each task must be printed. The following tasks describe the functionality required.

Task 4 [3 marks]

Write some code to find whether a given temperature exists in a list using a simple linear search. Do this without the use of functions.

Task 5 [1 mark]

Rewrite your solution for Task 4 to now be implemented as a function.

Task 6 [5 marks]

Write a function to sort a given list in order of increasing temperature and then print the sorted list.
Hint: *This task is easier to implement with functions. First, choose a sorting algorithm to implement and decompose it into functions. Make sure you implement this decomposition in Python to enable you to translate it.*

Task 7 [2 marks]

Write a function to find the median element in the list.
Hint: *Note that the median of a list of items is found by first sorting the list. Use your solution from Task 6 to do this.*

Task 8 [4 marks]

Solve Task 4 using a recursive implementation of binary search. ***Hint:*** *Remember that binary search requires a sorted list. Use your solution from Task 6 to do this.*