Learning objectives of this practical session

To be able to write MIPS programs involving decisions, while and for loops, lists, local variables, and functions.

Important:

- **Local variables must be stored on the runtime stack.**
- **Python and MIPS code are required for your Task 3 to be marked. However, Python solutions alone do not result in any marks.**
- **Both your Python and your MIPS code need to be properly commented.**
- **Use only instructions that appear in this year's MIPS reference sheet.**
- **Use the function names provided, as we will have testing harnesses for you to run that rely on the MIPS code for the functions to be labeled with those names.**

Checkpoint for the end of your lab in Week 3

To reach the check point (and thus pass the hurdle) you must complete Tasks 1 and 2, as well as the python code required for Task 3. Thus, the Moodle submission for the checkpoint should be a zip file containing the `task_1.asm`, `task_2.asm`, and `task_3.py` files. Make sure you submit before leaving the lab. Please remember that reaching the checkpoint is a **hurdle** and you will get a 0 if you do not reach it (read the pracGuide document for more details).

After the first 30 minutes of your lab in week 4, your demonstrator will interview you to assess your level of confidence with your code, and will asses its quality. Please make sure you zip all required files and submit them before leaving your lab. Otherwise, you will get a 0 for this prac.

Task 1 [10 marks]

Consider the following uncommented Python code:

```
a = int(input(”Enter integer:”)),
b = int(input(”Enter integer:”)),
if a > 0 and b > 0:
    print(a//2)
elif a == b or a < b:
    print(2*b)
else
    print(b//2)
```

which for a=b=3 prints 1 (a//2), for a=b=-8 prints -16 (2*b), for a=-7 and b=2 prints 4 (2*b), and for a=10 and b=-6 prints -3 (b//2).

Faithfully translate the above code into a commented MIPS program assuming both a and b are global variables, and store it in file `task_1.asm`. Test it with (at least) the above four examples, and ensure it runs properly. Make sure you **do not use** any **`mult`** or **`div`** instructions and, instead, you use shift instructions.

Task 2 [20 marks]

The following uncommented Python code reads a list of integers of the size given by the user and computes (and prints) the minimum element in the list:

```python
size = int(input("Enter list size: "))

the_list = [0] * size

for i in range(size):
    the_list[i] = int(input("Enter element "+ str(i) + ": "))

if size > 0:
    min = the_list[0]
    for i in range(1,size):
        item = the_list[i]
        if min > item:
            min = item

    print( "The minimum element in this list is " + str(min) + "\n ")
```

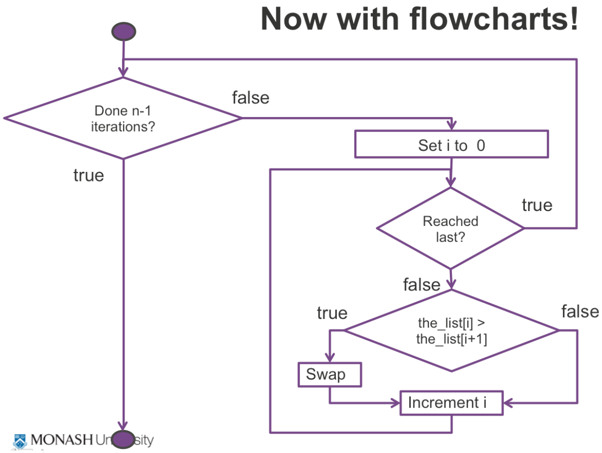
Faithfully translate the above code into a properly commented MIPS program assuming all variables are global, and store it in file `task_2.asm`. Ensure it runs properly.

Task 3 [15 marks]

(a) Write a Python program in file `task_3.py` that defines function `bubble_sort(the_list)`, which sorts `the_list` in increasing order, using the following algorithm (which is called "bubble sort"):

Assume the size of `the_list` is n. We will iterate $n-1$ times. In each of these iterations, we will start traversing `the_list` from position $i = 0$ and then do the following: while $i < n-1$

- Compare each element `the_list[i]` to the element on its right `the_list[i+1]`
- If `the_list[i]` > `the_list[i+1]` swap them, otherwise do not swap
- Increment i



We would recommend (but is not mandatory that) you add to your program a testing function `test_bubble_sort` that tests the function defined above with at least 3 examples.

(b) Write a MIPS program `task_3.asm` that implements `task_3.py` faithfully (no need to translate the test function).

Task 4 [15 marks]

Write in file `task_4.asm` a modification of your MIPS code in Task 2 to make it functional. That is, to faithfully translate the following Python code:

```python
def read_list():
    size = int(input("Enter list size: "))
    the_list = [0] * size

    for i in range(size):
        the_list[i] = int(input("Enter element "+ str(i) + ": "))
    return the_list

def get_minimum(the_list):
    size = len(the_list)
    if size > 0:
        min = the_list[0]
        for i in range(1,size):
            item = the_list[i]
            if min > item:
                min = item
        return min
    else:
        return 0 # this should return an exception (see later in the course)

if __name__ == '__main__':
    my_list = read_list()
    print(get_minimum(my_list))
```

You can copy and paste the part of the MIPS code you wrote in Task 2 that is useful for this one. However, **make sure your variables now are all local to the appropriate function (that is, no global variables are now allowed).**

Task 5 [20 marks]

Consider the following Python program:

```python
def frequency(item,the_list):
    counter = 0
    for element in the_list:
        if element == item:
            counter += 1
    return counter

if __name__ == '__main__':
    my_list = read_list()
    item = int(input("Enter the item: "))
    print(frequency(item,my_list))
```

where function `frequency(item,the_list)` returns how many times (if any) `item` appears in `the_list`, and `read_list()` is the function defined in the previous task. Write a MIPS program `task_5.asm` that faithfully translates the above code. Again, you can copy and paste code you defined before, and all variables must be local to the appropriate function.