AUT UNIVERSITY COMPUTER + MATHEMATICAL SCIENCES

FINAL COMPUTER-BASED EXAMINATION

Semester 1, 2015

PAPER DESCRIPTION: Programming 1/Programming for Engineering Applications

PAPER CODE: 405701/735318

TIME ALLOWED: **2.5** Hours plus **5** Minutes Reading Time

TOTAL MARKS: **120 marks**

INSTRUCTIONS:

1. **Read and follow the separately provided instructions sheet now**
2. **Do not start to read or answer questions until told to do so by an invigilator**
3. **This test has 12 questions, try to attempt them all**
4. **After completing the test you *must* sync before leaving. Do not leave until given permission to do so by an invigilator**

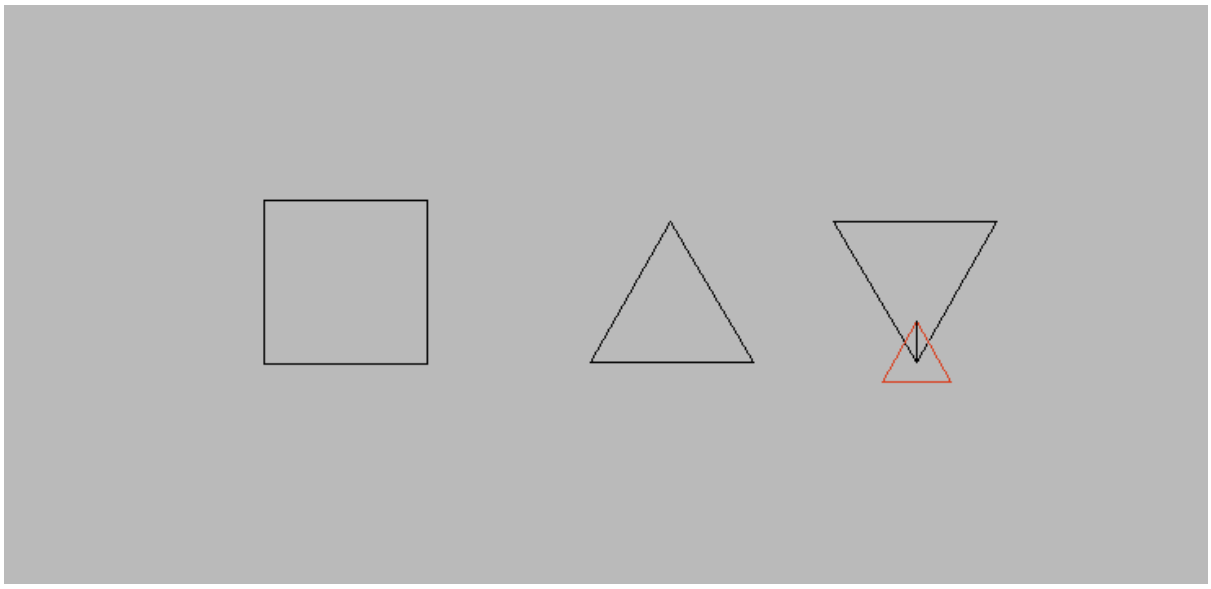
ADDITIONAL MATERIALS:

1. You may refer to your own notes and printed materials, including books
2. You may not use any electronic devices except for your Raspberry Pi. Turn off mobile phones
3. During the test you should not attempt in any way to communicate outside the exam room, except via sync

Q1 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q01`**

Complete the code in the file **`q01.c`** to draw the following.



The lengths of the sides of the triangles are 100. The inside angle of each point of a triangle is 60 degrees, meaning that the turtle must turn 120 degrees at each corner.

For full marks:

- Your code should be properly indented
- Leave the turtle in the position shown
- Call the function **`draw_triangle()`** from the function **`draw_inverted_triangle()`**

You can build your code using the following command

```
$ make
```

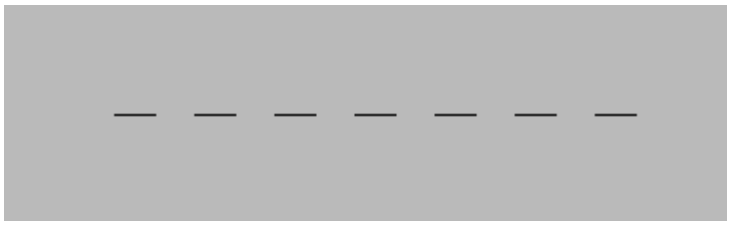
You can test your code using the following command

```
$ ./q01
```

Q2 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q02`**

Write code in **`q02.c`** to draw the dashed line shown below.



The dashes are each 20 pixels long. The spaces between them are also 20 pixels long. There are 7 dashes.

For full marks:

- The turtle should be made invisible before your program ends
- The start of the first dash should be the initial location of the turtle
- The end of the last dash should be the final location of the turtle
- Your code should use a loop
- Your code should be properly indented

Q3 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q03`**

Write code in **`q03.c`** to produce output exactly as shown below:

```
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q03$ ./q03
/--------------------------\
| Can you say "I love P1"? |
\--------------------------/
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q03$
```

For full marks:

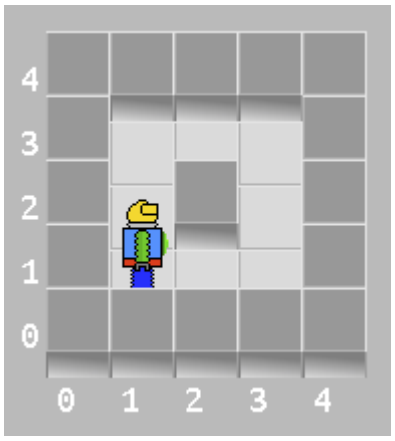
- Use no more than three calls to any standard library function
- Your code should *succeed*, i.e. the prompt should read **`[ok]`** after your code runs
- Your code should be properly indented

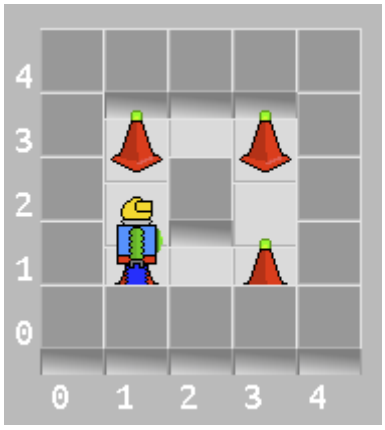
Q4 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q04`**

The robot starts in the south-west corner of a rectangular track. It must move around the track once returning to its starting point and orientation. It starts holding four beepers, and as it moves around the track it must place one at each corner. The track will vary in size between tests.

Here is the starting situation and target situation for the first test.





Implement this behaviour in a function called **`mark_corners()`** in the file **`q04.c`**.

You can compile and test your code using the command

```
$ make test
```

For full marks:

- All tests must pass
- Your algorithm should be able to handle a track of any dimensions
- Avoid repeated code
- Indent your code properly

Q5 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q05`**

In file **`q05.c`** the **`main()`** function defines two variables, **`a`** and **`b`**. Add code to **`main()`** to print output similar to what follows, giving the relevant details of those variables.

```
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q05$ ./q05
The address of a is 3198543348
The value of a is 17
The address of b is 3198543344
The value of b is 3198543348
The value of the variable that b points to is 17
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q05$
```

Hint: the directive **`%u`** is useful for printing memory addresses using the function **`printf()`**.

For full marks:

- Your code should be properly indented
- Your code should print actual runtime information regarding the variables, in the format given above

Q6 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q06`**

In **`q06.c`** write a function that can verify whether its parameters represent a valid time on a 24 hour clock, given in minutes and hours.

For full marks:

- Ensure all tests pass
- Implement your function using a single line of C in its body
- Give meaningful names to all variables
- Indent your code correctly

Q7 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q07`**

Write and compile a program to print the string **`Hello world!`**

For full marks

- Ensure that on submission the **`q07`** directory contains both your source code and a compiled executable
- Write a makefile that will allow your code to be compiled using only the command **`make`** without arguments.

Q8 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q08`**

In **`q08.c`** write a program that will open a text file given as the first argument on the command line and print its contents. For testing purposes the file **`test.txt`** is given. Output of the program when run on **`test.txt`** should be as follows.

```
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q08$ ./q08 test.txt
the quick
brown fox
jumped over
the lazy
dog
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q08$
```

For full marks:

- Your program should succeed if given a valid argument
- Your program should fail with return value **`1`** if no argument is given or the file cannot be opened
- Your program should not use the standard library functions **`printf()`** or **`fscanf()`**
- Give appropriate names to all functions and variables
- Indent your code properly

Q9 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q09`**

In **`q09.c`** write a program that will read a list of numbers from a text file and provide output as below. You can assume that a valid input file consists of positive and negative integers each on a separate line.

The first integer given will be the number of integers in the list and should not be considered to be a member of the list.

The program should:

- Print the length of the list (not including the first number in the file)
- Print the list in reverse
- Print the least number in the list
- Print the greatest number in the list

The files **`test1.txt`** and **`test2.txt`** are given for testing purposes. Output when applied to these files should be as follows:

```
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q09$ ./q09 test1.txt
10 numbers
92
61
36
11
-16
1
98
-17
65
-39
least: -39
greatest: 98
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q09$ ./q09 test2.txt
5 numbers
-32
72
92
61
-93
least: -93
greatest: 92
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q09$
```

For full marks:

- Your program should generate output exactly as shown
- Your program should work for lists of any size
- Your program should succeed for valid input files and fail with return value **1** if an argument is not given, the file cannot be opened, or contains unexpected contents
- Your code should use appropriate variable and function names
- Your code should be correctly indented

Q10 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q10`**

Complete the code in **`q10.c`** to generate the following output by calls to the function **`print_person()`**. Leave **`print_person()`** unmodified.

```
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q10$ ./q10
First name: Kim
Last name: Kardashian
First name of spouse: Kanye
Last name of spouse: West
First name: Kanye
Last name: West
First name of spouse: Kim
Last name of spouse: Kardashian
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q10$
```

For full marks

- Allocate any instances of **`struct person`** on the heap rather than the stack

Q11 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q11`**

The code in **`q11.c`** is intended to print the first 10 triangular numbers as follows:

```
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q11$ ./q11
1
3
6
10
15
21
28
36
45
55
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q11$
```

A triangular number is the sum of all of the positive integers up to a certain value. For example, the third triangular number is 6 = 1 + 2 + 3

The code in **`q11.c`** contains three distinct logical errors, which cause it to produce memory errors or incorrect output. One of these requires several lines of code to be added to the program. The other two can be fixed by modifying one existing line of code per problem.

Apply these fixes so that the program works properly, while retaining the overall structure of the algorithm given.

For full marks:

- The corrected program will generate the correct output as above
- Comments will identify each change, describing the problem and how it was fixed
- Minimal changes will have been made to the structure of the algorithm

Q12 [10 marks]

Navigate to **`~/p1.2015s1/exam/working_copy/q12`**

In the guess the number game one player things of a number between 0 and 100 and the other player has to guess it. In **`q12.c`** implement this game with the computer as the player who is guessing, and the human user is the player thinking of the number.

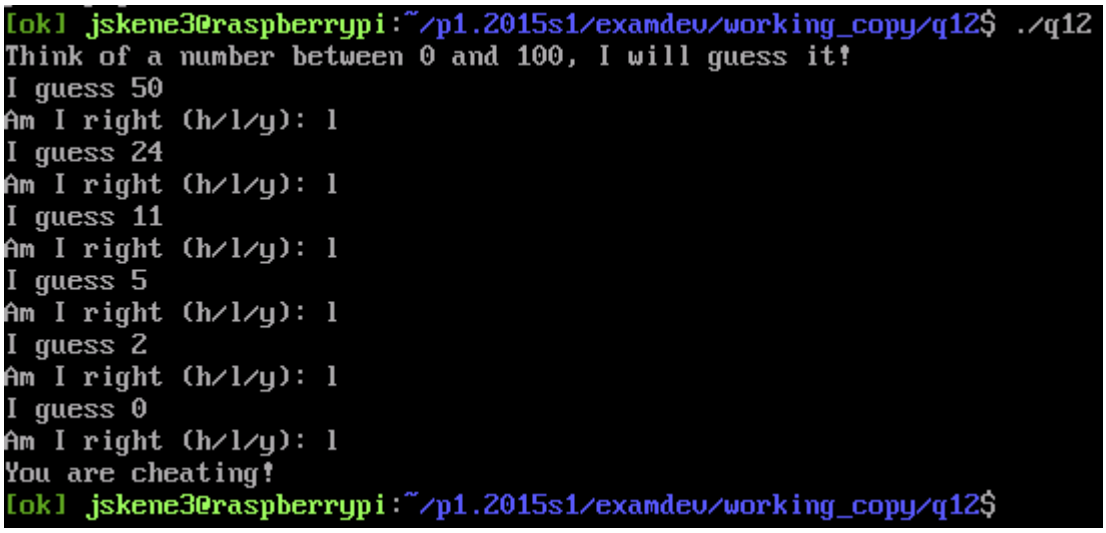
Here is a possible dialog if the user is thinking of the number 17:

```
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q12$ ./q12
Think of a number between 0 and 100, I will guess it!
I guess 50
Am I right (h/l/y): l
I guess 24
Am I right (h/l/y): l
I guess 11
Am I right (h/l/y): h
I guess 17
Am I right (h/l/y): y
Great! That's my favourite number too!
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q12$
```

Here is a possible dialog if the user is thinking of the number 16, and the computer is sure that must be the answer, having logically eliminated all other possibilities.

```
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q12$ ./q12
Think of a number between 0 and 100, I will guess it!
I guess 50
Am I right (h/l/y): l
I guess 24
Am I right (h/l/y): l
I guess 11
Am I right (h/l/y): h
I guess 17
Am I right (h/l/y): l
I guess 14
Am I right (h/l/y): h
I guess 15
Am I right (h/l/y): h
You must thinking of 16
[ok] jskene3@raspberrypi:~/p1.2015s1/examdev/working_copy/q12$
```

Here is a possible dialog if the user tries to cheat, leaving no possible number that could be the answer.



For full marks:

- Your program will be able to guess the number or detect cheating in less than 10 guesses
- Your code includes a comment explaining your guessing algorithm and stating why you believe it will be successful every time
- Your program will handle invalid inputs gracefully (for example, notifying the user and trying again)
- Your program will be structured into at least three non-trivial functions
- Your program will not use global variables
- It would be easy to modify your program so that the range of guesses was different from 0 and 100
- All names in your code are appropriate
- Your program is correctly indented

END OF EXAM