Exercise 0
Input two numbers and work out their sum, average and sum of the squares of the numbers.

Exercise 1
Input and output your name, address and age to an appropriate structure.

Exercise 2
Write a program that works out the largest and smallest values from a set of 10 inputted numbers.

Exercise 3
Write a program to read a "float" representing a number of degrees Celsius, and print as a "float" the equivalent temperature in degrees Fahrenheit. Print your results in a form such as
100.0 degrees Celsius converts to 212.0 degrees Fahrenheit.

Exercise 4
Write a program to print several lines (such as your name and address). You may use either several printf instructions, each with a newline character in it, or one printf with several newlines in the string.

Exercise 5
Write a program to read a positive integer at least equal to 3, and print out all possible permutations of three positive integers less or equal to than this value.

Exercise 6
Write a program to read a number of units of length (a float) and print out the area of a circle of that radius. Assume that the value of pi is 3.14159 (an appropriate declaration will be given you by ceilidh - select setup).
Your output should take the form: The area of a circle of radius ... units is .... units.
If you want to be clever, and have looked ahead in the notes, print the message Error: Negative values not permitted. if the input value is negative.

Exercise 7
Given as input a floating (real) number of centimeters, print out the equivalent number of feet (integer) and inches (floating, 1 decimal), with the inches given to an accuracy of one decimal place.
Assume 2.54 centimeters per inch, and 12 inches per foot.
If the input value is 333.3, the output format should be:
333.3 centimeters is 10 feet 11.2 inches.

Exercise 8

Given as input an integer number of seconds, print as output the equivalent time in hours, minutes and seconds. Recommended output format is something like 7322 seconds is equivalent to 2 hours 2 minutes 2 seconds.

Exercise 9

Write a program to read two integers with the following significance.
The first integer value represents a time of day on a 24 hour clock, so that 1245 represents quarter to one mid-day, for example.
The second integer represents a time duration in a similar way, so that 345 represents three hours and 45 minutes.
This duration is to be added to the first time, and the result printed out in the same notation, in this case 1630 which is the time 3 hours and 45 minutes after 12.45.
Typical output might be Start time is 1415. Duration is 50. End time is 1505.
There are a few extra marks for spotting.
Start time is 2300. Duration is 200. End time is 100.

Exercise 10

Write a program to read two characters, and print their value when interpreted as a 2-digit hexadecimal number. Accept upper case letters for values from 10 to 15.

Exercise 11 (hint: con il comando switch)

Read an integer value. Assume it is the number of a month of the year; print out the name of that month.

Exercise 12

Given as input three integers representing a date as day, month, year, print out the number day, month and year for the following day's date.
Typical input: 28 2 1992 Typical output: Date following 28:02:1992 is 29:02:1992

Exercise 13

Write a program which reads two integer values. If the first is less than the second, print the message up. If the second is less than the first, print the message down If the numbers are equal, print the message equal If there is an error reading the data, print a message containing the word Error and perform exit(0);

Exercise 14

Write a program to read in 10 numbers and compute the average (media), maximum and minimum values.

Exercise 15

Write a program to read in numbers until the number -999 is encountered. The sum of all number read until this point should be printed out.

Exercise 16

Write a program which will read an integer value for a base, then read a positive integer written to that base and print its value.
Read the second integer a character at a time; skip over any leading non-valid (i.e. not a digit between zero and ``base-1'') characters, then read valid characters until an invalid one is encountered.

Input	Output	
========	========	
10 1234	1234	
8 77	63	(the value of 77 in base 8, octal)
2 1111	15	(the value of 1111 in base 2, binary)

The base will be less than or equal to 10.

Exercise 17
Read in three values representing respectively
_a capital sum (integer number of pence),
_a rate of interest in percent (float),
_a number of years (integer).
Compute the values of the capital sum with compound interest added over the given period of years. Each year's interest is calculated as

interest = capital * interest_rate / 100;
and is added to the capital sum by capital += interest;

Print out money values as pounds (pence / 100.0) accurate to two decimal places.
Print out a floating value for the value with compound interest for each year up to the end of the period.
Print output year by year in a form such as:
Original sum 30000.00 at 12.5 percent for 20 years

Year	Interest	Sum
----	+-------	+--------
1	3750.00	33750.00
2	4218.75	37968.75
3	4746.09	42714.84
4	5339.35	48054.19
5	6006.77	54060.96
6	6757.62	60818.58
7	7602.32	68420.90
8	8552.61	76973.51
9	9621.68	86595.19
10	10824.39	97419.58

Exercise 18
Read a positive integer value, and compute the following sequence: If the number is even, halve it; if it's odd, multiply by 3 and add 1. Repeat this process until the value is 1, printing out each value. Finally print out how many of these operations you performed.

Typical output might be:

```
Inital value is 9
 Next value is  28
 Next value is  14
 Next value is   7
 Next value is  22
 Next value is  11
 Next value is  34
 Next value is  17
 Next value is  52
 Next value is  26
 Next value is  13
 Next value is  40
 Next value is  20
 Next value is  10
 Next value is   5
 Next value is  16
 Next value is   8
 Next value is   4
 Next value is   2
 Final value 1, number of steps 19
```

If the input value is less than 1, print a message containing the word "Error" and perform an exit(0); //return 0;

Exercise 19

Write a program to count the vowels and letters in free text given as standard input. Read text a character at a time until you encounter end-of-data.
Then print out the number of occurrences of each of the vowels a, e, i, o and u in the text, the total number of letters, and each of the vowels as an integer percentage of the letter total.
Suggested output format is:

```
Numbers of characters:
 a   3 ; e   2 ; i   0 ; o   1 ; u   0 ; rest  17
 Percentages of total:
 a  13%; e   8%; i   0%; o   4%; u   0%; rest  73%
```

Read characters to end of data using a construct such as

```
char ch;
 while((ch = getchar()) >= 0){
    /* ch is the next character */    ....
 }
```

to read characters one at a time using getchar() until a negative value is returned.

Exercise 20

Read a file of English text, and print it out one word per line, all punctuation and non-alpha characters being omitted.

For end-of-data, the program loop should read until "getchar" delivers a value <= 0. When typing input, end the data by typing the end-of-file character, usually control-D. When reading from a file, "getchar" will deliver a negative value when it encounters the end of the file.
Typical output might be
Read
a
file
of
English
text
and
print
it
out
one
...

Exercise 21
Write a C program to read through an array of any type. Write a C program to scan through this array to find a particular value.

Exercise 22
Read ordinary text a character at a time from the program's standard input, and print it with each line reversed from left to right. Read until you encounter end-of-data (see below).
You may wish to test the program by typing

prog5rev | prog5rev

to see if an exact copy of the original input is recreated.
To read characters to end of data, use a loop such as either

```
char ch;
while(ch=getchar(), ch >= 0 ) /* ch < 0 indicates end-of-data */
```

or

```
char ch;
 while( scanf( "%c", &ch ) == 1 ) /* one character read */
```

Exercise 23
Write a program to read English text to end-of-data (type control-D to indicate end of data at a terminal, see below for detecting it), and print a count of word lengths, i.e. the total number of words of length 1 which occurred, the number of length 2, and so on.
Define a word to be a sequence of alphabetic characters. You should allow for word lengths up to 25 letters.
Typical output should be like this:

length 1 : 10 occurrences
length 2 : 19 occurrences

length 3 : 127 occurrences
length 4 : 0 occurrences
length 5 : 18 occurrences
....

To read characters to end of data see above question.

Exercise 23

Write a function “replace'' which takes a pointer to a string as a parameter, which replaces all spaces in that string by minus signs, and delivers the number of spaces it replaced.

Thus

```
char *cat = "The cat sat";
 n = replace( cat );
```

should set

cat to "The-cat-sat"

and

n to 2.

Exercise 24

Write a program which will read in the source of a C program from its standard input, and print out all the starred items in the following statistics for the program (all as integers). (Note the comment on tab characters at the end of this specification.)

Print out the following values:

Lines:

* The total number of lines
* The total number of blank lines
  (Any lines consisting entirely of white space should be considered as blank lines.)
* The percentage of blank lines (100 * blank_lines / lines)

Characters:

* The total number of characters after tab expansion
* The total number of spaces after tab expansion
* The total number of leading spaces after tab expansion
  (These are the spaces at the start of a line, before any visible character; ignore them if there are no visible characters.)
* The average number of
  -characters per line
  -characters per line ignoring leading spaces
  -leading spaces per line
  -spaces per line ignoring leading spaces

Comments:

* The total number of comments in the program
* The total number of characters in the comments in the program
  excluding the "/*" and "*/" thenselves
* The percentage of number of comments to total lines
* The percentage of characters in comments to characters

Identifiers:

We are concerned with all the occurrences of "identifiers" in the program where each part of the text starting with a letter, and continuing with letter, digits and underscores is considered to be an identifier, provided that it is not in a comment, or in a string, or within primes.

Note that:

"abc\"def" -> the internal escaped quote does not close the string.

Also, the representation of the escape character is -> '\\'

and of prime is -> '\''

Do not attempt to exclude the fixed words of the language, treat them as identifiers. Print

* The total number of identifier occurrences.
* The total number of characters in them.
* The average identifier length.

Indenting:

* The total number of times either of the following occurs:
  a line containing a "}" is more indented than the preceding line
  a line is preceded by a line containing a "{" and is less
  indented than it.
  The "{" and "}" must be ignored if in a comment or string or
  primes, or if the other line involved is entirely comment.

A single count of the sum of both types of error is required.

NOTE: All tab characters ('') on input should be interpreted as multiple spaces using the rule:

"move to the next modulo 8 column"

where the first column is numbered column 0.

col before tab	col after tab
0	8
1	8
7	8
8	16
9	16
15	16
16	24

To read input a character at a time the skeleton has code incorporated to read a line at a time for you using

```
char ch;
 ch = getchar();
```

Which will deliver each character exactly as read. The "getline" function then puts the line just read in the global array of characters "linec", null terminated, and delivers the length of the line, or a negative value if end of data has been encountered.
You can then look at the characters just read with (for example)

```
switch( linec[0] ) {
 case ' ': /* space ..... */
        break;
 case '\t': /* tab character .... */
        break;
 case '\n': /* newline ... */
        break;
 ....
 } /* end switch */
```

End of data is indicated by scanf NOT delivering the value 1.
Your output should be in the following style:

Total lines	126
Total blank lines	3
Total characters	3897
Total spaces	1844
Total leading spaces	1180
Total comments	7
Total chars in comments	234
Total number of identifiers	132
Total length of identifiers	606
Total indenting errors	2

You may gather that the above program (together with the unstarred items) forms the basis of part of your marking system! Do the easy bits first, and leave it at that if some aspects worry you. Come back to me if you think my solution (or the specification) is wrong! That is quite possible!

Exercise 25
It's rates of pay again!
Loop performing the following operation in your program:
Read two integers, representing a rate of pay (pence per hour) and a number of hours. Print out the total pay, with hours up to 40 being paid at basic rate, from 40 to 60 at rate-and-a-half, above 60 at double-rate. Print the pay as pounds to two decimal places.

Terminate the loop when a zero rate is encountered. At the end of the loop, print out the total pay.
The code for computing the pay from the rate and hours is to be written as a function.
The recommended output format is something like:

Pay at 200 pence/hr for 38 hours is 76.00 pounds
Pay at 220 pence/hr for 48 hours is 114.40 pounds
Pay at 240 pence/hr for 68 hours is 206.40 pounds
Pay at 260 pence/hr for 48 hours is 135.20 pounds
Pay at 280 pence/hr for 68 hours is 240.80 pounds
Pay at 300 pence/hr for 48 hours is 156.00 pounds
Total pay is 928.80 pounds

The “program features'' checks that explicit values such as 40 and 60 appear only once, as a #define or initialised variable value. This represents good programming practice.

Exercise 26
Write program using enumerated types which when given today's date will print out tomorrow's date in the for 31st January, for example.

Exercise 27
Write a simple database program that will store a persons details such as age, date of birth, address ***etc***