Solutions to Assignment 1

Question 1

The steps involved in evaluating the C expressions are shown below.

1. ```
   2 * ' '
   2 * 32
   64
   ```

2. ```
   31 / 6 + 8
   5 + 8
   13
   ```

3. ```
   y = ( 1 + 2 ) == ( 4 - 1 )
   y = 3 == 3
   y = 1
   1
   ```

4. ```
   x = 1 == -2 + 3
   x = 1 == 1
   x = 1
   1
   ```

5. ```
   6 <= 4 + 9 / 2
   6 <= 4 + 4
   6 <= 8
   1
   ```

6. ```
   ( 0xab & 0x0f )
   0x0b
   ```

7. ```
   ( 0x2d & 0xf0 ) | ( 0x2d & 0x0f )
   0x20 | 0x0d
   0x2d
   ```

8. ```
   3 * ( 0x0d && 0xd0 )
   3 * 1
   3
   ```

9. ```
   ( 0x2d ^ 0xff ) + ( 1 << 1 )
   0xd2 + 2
   0xd4
   ```

10. ```
    ~ ( 256 | ' ' )
    ~ ( 0x0100 | 0x0020 )
    ~ 0x120
    0xfedf
    ```

    (all leading digits in the result are “F”)

11. ```
    128 || ( ' ' == 0x21 )
    128 || 0
    1
    ```

    (the second expression would actually be “short circuited” and would not be evaluated)

Question 2

The program will print the values of i:

```
2
4
5
6
8
```

Question 3

```
#include <stdio.h>
main()
{
   int i ;
   for ( i=6 ; i<=66 ; i = i+3 ) {
      if ( i >= 30 && i <= 45 ) {
        /* do nothing */
      } else {
        printf ( "%d\n", i ) ;
      }
   }
}
```
```

Question 4

```
int len( char s[] )
{
   int i ;
   i = 0 ;
   while ( s[i] != 0 ) {
      i++ ;
   }
   return i ;
}
```

Question 5

decimal	binary	hex
8	1000	0x8
7	0111	0x7
16	10000	0x10
15	1111	0xf
256	100000000	0x100
255	11111111	0xff
237	1110 1101	0xEd

Question 6

binary	hex	decimal
1011	0x0b	11
1011 1011	0xbb	187
1000 0000	0x80	128
11 1100	0x3c	60
0011 1100	0x3c	60

Question 7

hex	binary	decimal
0x0e	0000 1110	15
0xe	0000 1110	15
0xAA	1010 1010	170
0xFA	1111 1010	250
0x40	0100 0000	64
0x18	0001 1000	24

Question 8

The following code uses a mask that tests the bits in the argument in order from most- to least-significant bit. A '1' or a '0' character is printed depending on the bit's value.

```
void printbin ( int n )
{
   int i ;
   i = 16384 ;
   while ( i >= 1 ) {
      if ( i & n ) {
         printf ( "1" ) ;
      } else {
         printf ( "0" ) ;
      }
      i = i / 2 ;
   }
   printf ( "\n" ) ;
}
```

Question 9

The controller's two inputs are the on/off switch, `on-off`, and the temperature sensor, `temp`. The two outputs are the motor speed controls `slow` and `fast`.

The motor runs slowly when only the `slow` output is on. The motor runs fast when only the `fast` output is on. The motor is off when both outputs are off.

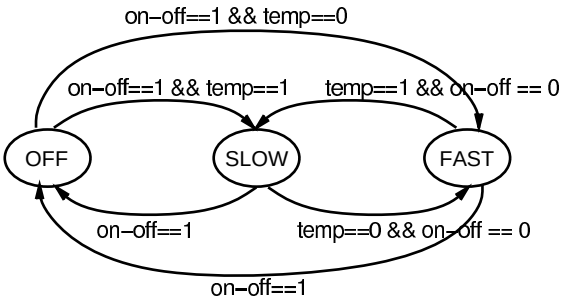
There are three combinations of outputs: both `off`, only `slow` on, and only `fast` on. We will tentatively choose three states and name them `SLOW`, `OFF`, and `FAST` corresponding to these combinations of outputs.

The outputs for each state are given in the table below:

state	slow	fast
OFF	0	0
SLOW	1	0
FAST	0	1

The following state transition diagram describes

the behaviour of the controller:



The following table also describes the state transitions:

starting state	input		next state
	`on-off`	`temp`	
OFF	0	X	OFF
OFF	1	0	FAST
OFF	1	1	SLOW
SLOW	1	X	OFF
SLOW	0	0	FAST
SLOW	0	1	SLOW
FAST	1	X	OFF
FAST	0	0	FAST
FAST	0	1	SLOW

If the button could be “pushed” for longer than one state transition then the machine would continue to cycle on and off while the operator held the button down.

To avoid this problem we could add additional states (for example, `OFF2`, `SLOW2` and `FAST2`) as the targets of state transitions where `on-off` is ’1’. The state machine would transition to the corresponding regular state (`OFF`, `SLOW` and `FAST`) when `on-off` returned to ’0’.

Another alternative would be to design a second state machine that operated in parallel with the first and that generated a momentary `on-off` signal that was only true for one state transition.