Halmstad University
Center for Research on Embedded Systems

Solutions to the Embedded Systems Programming Examination, April 2012

1.a.
a 5 and a 1

1.b.
a 5 and a 2 with an extra vertical bar closing a loop.

1.c.
a 5 and a 2.

2.a.
If KEY_STATUS_REG is an ordinary variable the program is an infinite loop. If it is an IO register, its value can change due to changes in the device to which it is associated, the loop would thus terminate: exit from the loop means that the content of the register has changed.

2b.
The program will the have to choose an order in which to test the status changes and might miss status changes in some devices while waiting for changes in other registers.

3.a.
There is only one global variable: pp. The only function that uses it is printAt.

3.b.
The function is executed in 2 different threads: the one that is computing primes starting from 0 and the one that is computing primes starting from 3. Here is an example of interleaving that results in an erroneous output:

```
thread 1            thread 2
...                 ...
pp = 0
writeChar('0',0)
                    pp = 3
p++
writeChar('2',4)
```

3.c.

tinythreads provides mutex variables with operations lock and unlock. In the program we add one global variable

```
mutex m = initMutex();
```

and the function printAt is changed to

```
lock(&m);
pp=pos;
writeChar
pp++
writeChar
unlock(&m)
```

4.

```
typedef struct{
  Object super;
  unsigned int * port;
} Port;
#define initPort(p) {initObject(),p}
int set(Port *self, unsigned int mask){
  *(self->port) = *(self->port) | mask;
}
int clear(Port *self, unsigned int mask){
  *(self->port) = *(self->port) & ~mask;
}
int toggle(Port *self, int bitNr){
  *(self->port) = *(self->port) ^ 1<<bitNr
}
```

5.

```
typedef struct{
  Object super;
  int running;
  int phase;
} Wave
#define initWave(d) = {initObject(),0,D}
int turnOn(Wave *self, int nothing){
  if(self->running == 0){
            self->running = 1;
            doWave(self,0);
  }
}
int turnOff(Wave *self, int nothing){
   self->running = 0;
}
int setPhase(Wave *self, int phase){
   self->phase = d;
}
int doAWave(Wave *self, int nothing){
  AFTER(MSEC(D), self, doBWave, 0);
  AFTER(MSEC(T), self, doAWave, 0);
  doA();
}
int doBWave(Wave *self, int nothing){
 doB();
}
void  doA(){//  to do what has to be done!}
void doB(){// to do what has to be done!}
```

6. a.
Components exchange data via Intents. The component that wants to start another component creates an Intent and adds data to it. The component that is started can read data from the Intent.

6.b.
Notifications can be used to let the user use other apps while an application is doing some task in the background. The latter can generate a notification that the user can use to reactivate the former app to see any expected results.