Problema 1 (3 puntos).

Explicar detalladamente (máximo un folio por una cara) como se comporta el siguiente programa:

```
#include <signal.h>

int pid;

int trata_alarma()
{
        kill(pid, SIGKILL);
}

main()
{
        int estado;

        pid = fork();
        if( pid ){
                signal(SIGALRM,trata_alarma);
                alarm(10);
                wait(&estado);
                alarm(0);
        }
        else{
                execl("/bin/p1","p1",0);
                exit(1);
        }
}
```

Problema 2 (3 puntos).

Diseñar un programa llamado "`crea`" en lenguaje C y usando sólo llamadas al sistema UNIX que actue como sigue:

- "`crea`" debe crear tantos hijos como se le indique en el primer parámetro pasado a "`crea`".
- cada uno de estos hijos debera enviar a su padre su identificador de proceso (PID) mediante una pipe y a continuación morirá.
- el padre debera escribir en un fichero (cuyo nombre es pasado como segundo parámetro a "`crea`") todos los mensajes que le lleguen (uno en cada linea).

Nota: Todos los hijos deberan utilizar la misma pipe.

Ejemplo: **merlin_$** `crea 15 hijos`

Creara 15 hijos cuyos identificadores aparecerán en el fichero "`hijos`".

Problema 3 (4 puntos).

Hemos escrito el siguiente programa en UNIX:

programa.c:

```
#include <fcntl.h>

int fdrd, fdwt;
char c;

main(argc,argv)
int argc;
char *argv[];
{
        if(argc != 3)   exit(1);
        if((fdrd = open(argv[1],O_RDONLY)) == -1)
                exit(1);
        if((fdwt = creat(argv[2],0666)) == -1)
                exit(1);
        fork();
        rdwrt();
        exit(0);
}

rdwrt()
{
        for(;;)
        {
                if(read(fdrd,&c,1) != 1)
                        return;
                write(fdwt,&c,1);
        }
}
```

Explica cuál será el resultado de la ejecución de "`programa`" y por qué.