Material permitido: **Calculadora NO programable.** Tiempo: **2 horas.** N1	**Aviso 1:** Todas las respuestas deben estar razonadas. **Aviso 2:** Escriba sus respuestas con una letra **lo más clara posible.** **Aviso 3: No use *Tipp-ex*** o similares (atasca el escáner).

ESTE EXAMEN CONSTA DE 5 PREGUNTAS

1. (*2 p*) Conteste razonadamente si las siguientes afirmaciones son verdaderas o falsas:

a) (*1 p*). La llamada al sistema `lseek` tiene tres parámetros de entrada. Uno de ellos indica el número de bytes que se va a desplaza el puntero de lectura/escritura. Por tanto, dicho parámetro solo puede ser un número entero positivo.

b) (*1 p*) Para transferir la abstracción de hebra enteramente al nivel de usuario, debe intervenir el núcleo.

2. (*1.5 p*) Describe qué son las variables de condición y para qué se utilizan.

3. (*1.5 p*) Conteste razonadamente las siguientes cuestiones:

a) *(0.5 p)* ¿Para qué usa el núcleo el algoritmo `syscall()`?

b) (*1.5 p*) Dibuje un diagrama adecuadamente rotulado que resuma las principales acciones que realiza este algoritmo.

4. *(2 p)* Considérese una cache de buffers de bloques en un determinado sistema de ficheros que utiliza una sola partición. Esta cache consta de cuatro colas de dispersión. En un cierto instante de tiempo *t*, esta cache contiene una copia de 12 bloques de disco, cuyos números son: 11, 74, 85, 32, 8, 13, 2, 31, 41, 56, 47, 82. Dibuje un posible esquema adecuadamente rotulado de la cache de buffers de bloques en dicho instante t.

5. *(3 p)* Conteste razonadamente a los siguientes apartados:

a) *(1.5 p)* Explicar el significado de las sentencias enumeradas (**[ ]**) del programa que se muestra en la página siguiente.

b) *(1.5 p)* Si el programa anterior se compila produciendo el ejecutable DyASO y se ejecuta con la orden `./DyASO`. Explique detalladamente el funcionamiento del programa así como su salida suponiendo que el primer proceso creado tiene pid=1000, los pids de los procesos se asignan consecutivamente y no se produce ningún error.

NOTA: la función `snprintf (buffer_de_salida, tamaño_del_buffer, cadena_de_control, Arg1, … ArgN )` formatea una cadena de texto usando una cadena de control y un conjunto de argumentos del mismo modo que `printf`. Pero, en lugar de escribirlo en la salida estándar, lo hace en el buffer que se le pasa como primer argumento y limitando la salida (si fuese necesario) al tamaño especificado en el segundo argumento.
NOTA2: La función `fflush(stdout)` obliga a que la escritura por salida estándar se realice inmediatamente impidiendo condiciones de carrera en la impresión del buffer de salida estándar `stdout`.

La pregunta continua en la página siguiente

```
        #include <stdio.h>
        #include <stdlib.h>
        #include <string.h>
        #include <unistd.h>
        #include <sys/msg.h>
[1]     void main(int argc, char *argv[])
        {
        key_t llave;
        struct{ long tipo;
             char cadena[2];
             } mensaje;
        int i, pid, msqid, N, longitud=sizeof(mensaje)-sizeof(mensaje.tipo);
        char Ntexto[10], texto[]="Ejercicio 5 DyASO ";
[2]     if ( (llave=ftok(argv[0],'M'))==-1) {perror("Error ftok"); exit(-1); }
[3]     if ( (msqid=msgget(llave, IPC_CREAT | 0600))==-1 )
                             {perror("Error msgget"); exit(-2); }

        if (argc>2) {printf("Argumentos incorrectos\n"); exit(-3); }
        if (argc==1) {
             N=strlen(texto);
             for(i=0; i<=N; i++) {
                  mensaje.tipo=1;
                  mensaje.cadena[0]=texto[i];
                  mensaje.cadena[1]=0;
[4]               msgsnd(msqid, &mensaje,longitud,0);
                  }
             }
        else      N=atoi(argv[1]);

[5]     if ( (pid=fork())==-1 ) {perror("Error fork"); exit(-4);}
        if (pid==0) {
             if (N>0) {
                  printf("Proceso hijo %d\n",getpid());
                  fflush(stdout);
                  snprintf(Ntexto, 10 , "%d",N-1);
[6]               if(execl(argv[0],argv[0],Ntexto,NULL)==-1)
                        {perror("Error en execl"); exit(-4);}
                  }
             printf("Fin ultimo hijo %d\n",getpid());
             fflush(stdout);
[7]          sleep(1);
             }
        else {
             wait();
[8]          msgrcv(msqid, &mensaje,longitud,1,0);
[9]          printf("%s",mensaje.cadena);
             fflush(stdout);
             if (argc==1) printf("correcto\n");
             }
        }
```