Problema 1:

- ¿Cuantos procesos se crean en el siguiente código (incluido el proceso inicial)? Dibuja el árbol de procesos

```
#include <stdio.h>
#include <unistd.h>

int main() {
    /* fork a child process */
    fork();
    /* fork another child process */
    fork();
    /* and fork another */
    fork();
    return 0;
}
```

Problema 2:

- ¿Cuantos procesos se crean en el siguiente código (incluido el proceso inicial)? Dibuja el árbol de procesos

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int n=4;

int main() {
    pid_t pid;
    int i;
    for (i=0; i<n ; i++) {
        pid = fork();
        if (pid > 0) {
            pid = wait(NULL);
            exit(0);
        }
    }
}
```

Problema 3:

- Determina los valores de pid que se imprimen en A, B, C y D (supón que los pids del padre y el hijo son 2600 y 2603 respectivamente)

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main() {
    pid_t pid, pid1;
    pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork failed\n");
        return -1;
    } else if (pid == 0) {
        pid1 = getpid();
        printf("child: pid = %d\n", pid);   /* A */
        printf("child: pid1 = %d\n", pid1); /* B */
    } else {
        pid1 = getpid();
        printf("parent: pid = %d\n", pid);   /* C */
        printf("parent: pid1 = %d\n", pid1); /* D */
        wait(NULL);
    }
    return 0;
}
```

Problema 4:

- Determina el valor que se imprime en la línea A

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;

int main() {
    pid_t pid;
    pid = fork();
    if (pid == 0) { /* child process */
        value += 15;
        return 0;
    } else if (pid > 0) { /* parent process */
        wait (NULL);
        printf("parent: value = %d\n", value); /* A */
        return 0;
    }
}
```

Problema 5:

- Determina el valor que se imprime en las líneas A y B

```
#include <pthread.h>
#include <stdio.h>
void *runner(void *param);

int value = 5;

int main() {
    pid_t pid;
    pthread_t tid;
    pthread_attr_t attr;

    pid = fork();
    if (pid == 0) { /* child process */
        pthread_attr_init(&attr);
        pthread_create(&tid, &attr, runner, NULL);
        pthread_join(tid, NULL);
        printf("child: value = %d\n", value); /* A */
    } else if (pid > 0) { /* parent process */
        wait (NULL);
        printf("parent: value = %d\n", value); /* B */
    }
}

void *runner(void *param) {
    value += 15;
    pthread_exit(0);
}
```