The first Project topic: Dining Philosophers

To solve this problem, student can use "C codes" described at below web page. But, students in oral exam will be responsible for,

- flow of the solution,
- task performed by each procedure,
- any changing request for the source codes.

http://rosettacode.org/wiki/Dining_philosophers#C

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdarg.h>

#define N 5
const char *names[N] = { "Aristotle", "Kant", "Spinoza", "Marx", "Russell" };
pthread_mutex_t forks[N];
#define M 5 /* think bubbles */
const char *topic[M] = { "Spaghetti!", "Life", "Universe", "Everything", "Bathroom" };

#define lock pthread_mutex_lock
#define unlock pthread_mutex_unlock
#define xy(x, y) printf("\033[%d;%dH", x, y)
#define clear_eol(x) print(x, 12, "\033[K")

void print(int y, int x, const char *fmt, ...)
{
        static pthread_mutex_t screen = PTHREAD_MUTEX_INITIALIZER;
        va_list ap;
        va_start(ap, fmt);
        lock(&screen);
        xy(y + 1, x), vprintf(fmt, ap);
        xy(N + 1, 1), fflush(stdout);
        unlock(&screen);
}

void eat(int id)
{
        int f[2], ration, i; /* forks */
        f[0] = f[1] = id;
        /* make some (but not all) philosophers leftie.
           could have been f[!id] = (id + 1) %N; for example */
        f[id & 1] = (id + 1) % N;
        clear_eol(id);
        print(id, 12, "..oO (forks, need forks)");
        for (i = 0; i < 2; i++) {
                lock(forks + f[i]);
                if (!i) clear_eol(id);

                print(id, 12 + (f[i] != id) * 6, "fork%d", f[i]);
                /* delay 1 sec to clearly show the order of fork acquisition */
                sleep(1);
        }
        for (i = 0, ration = 3 + rand() % 8; i < ration; i++)
                print(id, 24 + i * 4, "nom"), sleep(1);
        /* done nomming, give up forks (order doesn't matter) */
        for (i = 0; i < 2; i++) unlock(forks + f[i]);
}

void think(int id)
{
        int i, t;
        char buf[64] = {0};
        do {
                clear_eol(id);
                sprintf(buf, "..oO (%s)", topic[t = rand() % M]);
                for (i = 0; buf[i]; i++) {
                        print(id, i+12, "%c", buf[i]);
                        if (i < 5) usleep(200000);
                }
                usleep(500000 + rand() % 1000000);
```

```
        } while (t);
}

void* philosophize(void *a)
{       int id = *(int*)a;
        print(id, 1, "%10s", names[id]);
        while(1) think(id), eat(id);
}

int main()
{       int i, id[N];
        pthread_t tid[N];
        for (i = 0; i < N; i++)
                pthread_mutex_init(forks + (id[i] = i), 0);
        for (i = 0; i < N; i++)
                pthread_create(tid + i, 0, philosophize, id + i);
        /* wait forever: the threads don't actually stop */
        return pthread_join(tid[0], 0);
}
```