# Drugi kolokvijum iz Operativnih sistema 1
# Jun 2022.

**1. (10 poena)**

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>

#define handle_error(msg) do { \
  printf(msg); \
  exit(-1); \
} while (0)

const int M = ..., N = ...;
extern double mat[M][N];

int max (int i) {
  double m = mat[i][0];
  int mj = 0;
  for (int j=1; j<N; j++)
    if (mat[i][j]>m)
      m = mat[i][j], mj = j;
  return mj;
}

pid_t pids[M];

int main () {

  for (int i=0; i<M; i++) {
    pid_t pid = pids[i] = fork();
    if (pid<0) handle_error("Error: Cannot create a child process.\n");
    if (pid==0)
      exit(max(i));
  }

  double max;
  int mj;
  if (waitpid(pids[0],&mj)<0)
    handle_error("Error waiting a child process.\n");
  max = mat[0][mj];

  for (int i=1; i<M; i++) {
    if (waitpid(pids[i],&mj)<0)
      handle_error("Error waiting a child process.\n");
    if (mat[i][mj]>max) max = mat[i][mj];
  }
  printf("Max: %f\n",max);
  exit(0);
}
```

**2. (10 poena)**

```
class Condition {
public:
  Condition (bool init = false) : cond(init) {}

  void set ();
  void clear () { lock(); cond = false; unlock(); }
  void wait ();

private:
  bool cond;
  Queue blocked;
};

void Condition::wait () {
  lock();
  if (!cond)
    if (setjmp(Thread::runningThread->context)==0) {
      blocked.put(Thread::runningThread);
      Thread::runningThread = Scheduler::get();
      longjmp(Thread::runningThread->context,1);
    }
  unlock();
}


void Condition::set () {
  lock();
  cond = true;
  for (Thread* t = blocked.get(); t; t = blocked.get())
    Scheduler::put(t);
  unlock();
}
```

**3. (10 poena)**

```
#include "kernel.h"

class SharedCoord {
public:
  SharedCoord ();

  void read (int& x, int& y);
  void write (int x, int y);

private:
  Semaphore mutex;
  int x, y;
};

inline SharedCoord () : mutex(1) {}

inline void SharedCoord::read (int& x_, int& y_) {
  mutex.wait();
  x_ = this->x;
  y_ = this->y;
  mutex.signal();
}

inline void SharedCoord::write (int x_, int y_) {
  mutex.wait();
  this->x = x_;
  this->y = y_;
  mutex.signal();
}
```