Rešenja zadataka sa trećeg kolokvijuma iz Operativnih sistema 2, januar 2019.

1. (10 poena)

```
class CLook {
public:
  CLook () : pending(0) {}

  Req* get () const { return pending; }
  void remove ();
  void put (Req*);

private:
  Req *pending;
};

void CLook::remove () {
  if (pending==0) return;
  if (pending->next==pending) {
    pending->next = pending->prev = 0;
    pending = 0;
  } else {
    Rq* rq = pending;
    pending = pending->next;
    rq->prev->next = pending;
    rq->next->prev = rq->prev;
    rq->next = rq->prev = 0;
  }
}
```

2. (10 poena)

```
freqs=$(cat /proc/cpuinfo | grep MH | cut -d: -f2 | tr -d " " | \
      cut -d. -f1)
sum=0
for i in $freqs; do
      let sum=sum+i
done
count=$(echo $freqs | wc -w)
if [ $count -gt 0 ]; then
      let average=sum/count
      echo "Average=$average"
fi
```

3. (10 poena)

```
class FileTree : public Tree{
public:
    FileTree(void *startAddress, long size) : Tree(startAddress, size) {}

    void *getLogicalAddress(void *address) override {
        uint64_t start, curr, result;

        start = (uint64_t) segmentStartAddress;
        curr = (uint64_t) address;

        if (curr < start || curr > start + segmentSize) {
            return 0;
        }

        result = curr - start;
        return (void *) result;
    }
```

```
    void *getVirtualAddress(void *address) override {
        uint64_t start, curr, result;

        start = (uint64_t) segmentStartAddress;
        curr = (uint64_t) address;

        if (curr > start) {
            return 0;
        }

        result = curr - start;
        return (void *) result;
    }
};

#define MEMORY_SIZE (10000)
#define N (100)

int main(int argc, char* argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Arguments error");
        exit(1);
    }
    int file = open(argv[1], O_RDWR | O_CREAT, S_IRUSR | S_IWUSR | S_IRGRP
| S_IWGRP);
    void  *tree_space  =  mmap(0,  MEMORY_SIZE,  PROT_READ  |  PROT_WRITE,
MAP_SHARED, file, 0);

    // If file content does not exists
    // ftruncate(file, MEMORY_SIZE);

    FileTree tree(tree_space, MEMORY_SIZE);
    for (int i = 0; i < N; i++) {
        tree.addNumber(rand());
    }

    munmap(tree_space, MEMORY_SIZE);
    close(file);

    return 0;
}
```