1.

Time	HRRN	FIFO/FCFS	RR	SJF	Priority
1	A	A	A	A	A
2	A	A	A	A	B
3	A	A	B	A	A
4	A	A	A	A	D
5	B	B	D	B	D
6	D	D	A	D	C
7	D	D	C	D	C
8	C	C	D	C	C
9	C	C	C	C	A
10	C	C	C	C	A
Avg. Turn-around Time	4.5	4.5	4.75	4.5	4.25

2.

Design idea

- Add a syscall interface `sys_setgood` in **user/libs/syscall.c**. The interface will use inline asm to call `ecall` to switch to **kernel mode**.
- Also add a syscall `sys_setgood` in **kern/syscall**, this syscall will call **do_setgood** in proc.c.
- In **do_setgood**, change the `labschedule_good` and `need_resched`.
- Change the `pick_next` in `default_sched.c`. When the scheduler needs to switch another process, it selects the process with max labschedule_good.

Modified code

```
int
set_good(int labschedule_good) {
    cprintf("set good to %d\n", labschedule_good);
    return sys_setgood(labschedule_good);
}
```

```c
int
sys_setgood(int64_t labschedule_good) {
    return syscall(SYS_setgood, labschedule_good);
}
```

```c
// ass04
#define SYS_setgood          16
```

```c
static int sys_setgood(uint64_t arg[]) {
    int labschedule_good = (int) arg[0];
    return do_setgood(labschedule_good);
}
```

```c
static int (*syscalls[])(uint64_t arg[]) = {
    [SYS_exit]              sys_exit,
    [SYS_fork]              sys_fork,
    [SYS_wait]              sys_wait,
    [SYS_exec]              sys_exec,
    [SYS_yield]             sys_yield,
    [SYS_kill]              sys_kill,
    [SYS_getpid]            sys_getpid,
    [SYS_putc]              sys_putc,
    [SYS_gettime]           sys_gettime,

    [SYS_setgood]           sys_setgood,
};
```

```c
// ass04
int
do_setgood(int labschedule_good) {
    current->labschedule_good = labschedule_good;
    current->need_resched = 1;
}
```

```c
static struct proc_struct *
RR_pick_next(struct run_queue *rq) {
    list_entry_t *le = list_next(&(rq->run_list));
    list_entry_t *cur = &(rq->run_list);

    struct proc_struct * nextProc = le2proc(le, run_link);
    struct proc_struct * proc;

    while (le != cur) {
        proc = le2proc(le, run_link);
        // 大的先运行
        if (proc->labschedule_good > nextProc->labschedule_good || \
        (nextProc->labschedule_good == proc->labschedule_good && proc->pid < nextProc->pid)) {
            nextProc = proc;
        }

        // 下一个
        le = list_next(le);
    }

    return nextProc;
}
```

Running sequence

```
1 2
3 4 5 6 7
6
2 5
2 3
2 7
2 4
2
1
```

Running result

```
memory management: default_pmm_manager
physcial memory map:
  memory: 0x08800000, [0x80200000, 0x885fffff].
sched class: RR_scheduler
SWAP: manager = fifo swap manager
setup timer interrupts
The next proc is pid:1
The next proc is pid:2
kernel_execve: pid = 2, name = "ex3".
Breakpoint
main: fork ok,now need to wait pids.
The next proc is pid:3
set good to 3
The next proc is pid:4
set good to 1
The next proc is pid:5
set good to 4
The next proc is pid:6
set good to 5
The next proc is pid:7
set good to 2
The next proc is pid:6
child pid 6, acc 4000001
The next proc is pid:2
The next proc is pid:5
set good to 4
child pid 5, acc 4000001
The next proc is pid:2
The next proc is pid:3
set good to 3
child pid 3, acc 4000001
The next proc is pid:2
The next proc is pid:7
child pid 7, acc 4000001
The next proc is pid:2
The next proc is pid:4
```

```
child pid 4, acc 4000001
The next proc is pid:2
main: wait pids over
The next proc is pid:1
all user-mode processes have quit.
The end of init_main
kernel panic at kern/process/proc.c:413:
    initproc exit.
```