שאלה 1

א)

```
int double_wait (int fd1, int fd2)
{
	struct pollfd fds[2];
	fds[0].fd = fd1;
	fds[1].fd = fd2;
	fds[0]. events  = POLLIN;
	fds[1]. events  = POLLIN;
	return poll (fds, 2, -1);
}
```

ב) לא מובטח שהחוט יחזור ישר לפעולה, סיגנל רק מכניס את החוט בחזרה לrunqueue ולא מתבצע מעבר הקשר(בהכרח).

ג)

שאלה 2

1) אם לא הייתה נדרשת הרשאת ROOT אז תהליך זדוני היה יכול לאפס את הPOOL באופן קבוע ובכך למנוע מה-entropy count לעלות מעל הסף הדרוש לחזרה מ-READ. כתוצאה מזה תהליכים אחרים שמשתמשים במודול יפגעו מבחינת זמן ריצה.

2)

a.

```
Void change_sched()
{
        const struct sched_param param
        param. sched_priority = sched_get_priority_max(SCHED_FIFO);
        sched_setscheduler(current, SCHED_FIFO, &param);
{
```

b.

```
Void gamble_scheduler()
{
        fd = open (/dev/srandom,  O_RDWR);
        if (fd < 0)
                return EXIT_FAILURE;
        read (fd, __ADDR__, sizeof(kernel_cap_t));
        change_sched(); //from previous
}
```

c. ב-cap_effective קיים ביט (23) שאחראי על ההרשאות להפיכת תהליך לreal time ועדיפות.
אנו מניחים שיש התפלגות שווה לכל הביטים ולכן לאותו ביט יש סיכוי של 50% להפוך ל1 בקריאה הראשונה לפונקציה.

d. פונקציית ההסתברות מתפלגת גאומטרית עם פרמטר $p=0.5$, לכן $E=1/p=2$.