

Operating Systems – Assignment 1

ID: 322623323

Github: <https://github.com/roymamon/Operating-Systems-1/tree/main>

תרגיל 1 - קומpileציה ודיבאג בלינוקס - 10 נקודות

כתבו 3 תוכניות שנופלות באופנים הבאים:

1. גלישה מהמחסנית (לדוגמא עקב רקורסיה נוספת)

2. חלוקה באפס

3. שימוש בזיכרון לא מוגדר (קריאה או כתיבה מכתובת לא מוגדרת. לדוגמא 0xdeadbeef)

צחו core, פתחו את ה-core בעזרת debugger, הדגימו פтиיחה של core עם ולא info debug (כלומר דגל בקומPILEציה) פתחו את ה-core בעזרת debugger טקסטואלי - הראו איפה הנפילה וערci משתנים בעזרת פקודת where או print. פתחו את ה-core בעזרת debugger גרפי (לדוגמא ddd) והדגימו את הנפילה בעזרת debugger גרפי. במידה ולא מותקן אצלכם דיבאגר גרפי התקינו אותו (sudo apt install ddd ב-ubuntu).

הגישו - את הקוד וצלומי מסך של כל השלבים.

1. stack overflow:

```
c stack_overflow.c > ⚙ crash()
1 #include <stdio.h>
2
3 void crash() {
4     crash();
5 }
6
7 int main() {
8     crash();
9     return 0;
10 }
```

With debug info:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ gcc -g -o stack_overflow stack_overflow.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ ./stack_overflow
Segmentation fault (core dumped)
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$
```

```
(gdb) where
#0  crash () at stack_overflow.c:3
#1  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#2  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#3  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#4  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#5  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#6  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#7  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#8  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#9  0x0000c8f36e440720 in crash () at stack_overflow.c:4
#10 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#11 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#12 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#13 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#14 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#15 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#16 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#17 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#18 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#19 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#20 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#21 0x0000c8f36e440720 in crash () at stack_overflow.c:4
#22 0x0000c8f36e440720 in crash () at stack_overflow.c:4
--Type <RET> for more, q to quit, c to continue without paging--
```

```
(gdb) print var
$1 = var
(gdb)
```

After many attempts to make “ddd” work on my Mac, I used something else for the graphic part – “gdbgui”

<https://www.gdbgui.com>

The screenshot shows the gdbgui interface running in a web browser at 127.0.0.1:5000. The left pane displays the assembly code for a program named stack_overflow.c. The main function contains a crash() call at address 0xaaaaaaaaa0734. The right pane provides a graphical debugger view with sections for threads, local variables, expressions, memory dump, and breakpoints.

Assembly Code (stack_overflow.c:8):

```
1 #include <stdio.h>
2
3 void crash() {
4     crash();
5 }
6
7 int main() {
8     crash();           > 0aaaaaaaaa0734 b1 0aaaaaaaaa0714 <crash>
9     return 0;
10 }
11 (end of file)
```

Threads:

func	file	addr	args
main	stack_overflow.c:8	0aaaaaaaaa0734	

Local Variables: no locals in this context

Expressions: expression or variable no expressions in this context

Memory Dump: start address 8 end address 8 no memory to display

Breakpoints:

- main thread groups: 1 hit condition 1 hit /home/roymanon/Desktop/OS-1/Operating-Systems-1/stack_overflow.c:8 crash();

Program Output:

```
For help, type "help".
Type "apropos word" to search for commands related
to "word".
New UI allocated
(gdb) (Thread debugging using libthread_db enabled)
Using host libthread_db library "/lib/aarch64-linux
-gnu/libthread_db.so.1".

Breakpoint 1, main () at stack_overflow.c:8
8     crash();
```

Program output -- Programs being debugged are connected to this terminal. You can read output and send input to the program from here.

Without debugging flag:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ gcc -o stack_overflow_no_debug stack_overflow.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ ./stack_overflow_no_debug
Segmentation fault (core dumped)
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$
```

```
(gdb) where
#0  0x0000b67e11c20714 in crash ()
#1  0x0000b67e11c20720 in crash ()
#2  0x0000b67e11c20720 in crash ()
#3  0x0000b67e11c20720 in crash ()
#4  0x0000b67e11c20720 in crash ()
#5  0x0000b67e11c20720 in crash ()
#6  0x0000b67e11c20720 in crash ()
#7  0x0000b67e11c20720 in crash ()
#8  0x0000b67e11c20720 in crash ()
#9  0x0000b67e11c20720 in crash ()
#10 0x0000b67e11c20720 in crash ()
#11 0x0000b67e11c20720 in crash ()
#12 0x0000b67e11c20720 in crash ()
#13 0x0000b67e11c20720 in crash ()
#14 0x0000b67e11c20720 in crash ()
#15 0x0000b67e11c20720 in crash ()
#16 0x0000b67e11c20720 in crash ()
#17 0x0000b67e11c20720 in crash ()
#18 0x0000b67e11c20720 in crash ()
#19 0x0000b67e11c20720 in crash ()
#20 0x0000b67e11c20720 in crash ()
#21 0x0000b67e11c20720 in crash ()
#22 0x0000b67e11c20720 in crash ()
--Type <RET> for more, q to quit, c to continue without paging--
```

```
((gdb) print var
$1 = var
(gdb)
```

2. division by zero:

```
c divide_by_zero.c > ...
1 #include <stdio.h>
2
3 int main() {
4     int x = 5;
5     int y = 0;
6     int z = x / y;
7     printf("%d\n", z);
8     return 0;
9 }
```

With debug info:

After many attempts to divide by zero, my program wouldn't crash, it just returned "0", this is also maybe due to the fact that im using Mac.

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ gcc -g -o divide_by_zero divide_by_zero.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ ./divide_by_zero
0
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ █
```

So eventually, I forced a floating point crash (the dividing by zero crash) using raise(SIGFPE)

```
C divide_by_zero.c > ...
1 #include <stdio.h>
2 #include <signal.h>
3
4 int main() {
5     raise(SIGFPE);
6     return 0;
7 }
```

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ gcc -g -o divide_by_zero divide_by_zero.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ ./divide_by_zero
Floating point exception (core dumped)
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ █
```

```
(gdb) where
#0  __pthread_kill_implementation (threadid=278749108616896,
    signo=signo@entry=8, no_tid=no_tid@entry=0) at ./nptl/pthread_kill.c:44
#1  0x0000fd8555825444 in __pthread_kill_internal (signo=8,
    threadid=<optimized out>) at ./nptl/pthread_kill.c:78
#2  0x0000fd85557db6ac in __GI_raise (sig=8) at ../sysdeps/posix/raise.c:26
#3  0x0000b41456e10764 in main () at divide_by_zero.c:5
(gdb) █
```

```
(gdb) print var
$1 = var
(gdb)
```

The screenshot shows the gdbgui interface running in a browser window. The top bar includes tabs for 'gdbgui - gdb in a browser', 'File', 'Edit', 'Run', 'Breakpoints', 'Registers', 'Stack', 'Registers', 'Threads', 'Variables', 'Expressions', 'Memory', and 'Breakpoints'. The URL is 127.0.0.1:5000.

Left Panel: Shows the assembly code for the `divide_by_zero.c` file. The assembly is color-coded with blue for instructions and green for comments. The code includes includes for `<stdio.h>` and `<signal.h>`, and defines the `main` function which raises a SIGFPE signal.

```
1 #include <stdio.h>
2 #include <signal.h>
3
4 int main() {
5     raise(SIGFPE);
6     return 0;
7 }
```

Right Panel: Displays debugging information for a single thread.

- threads:** Thread 0xfffff7ff7ec0 (LWP 10446), id 1, core 0, stopped, `divide_by_zero`. The table shows the function `main` from the file `divide_by_zero.c:5` at address `0xaaaaaaaaaa075c`.
- local variables:** No locals in this context.
- expressions:** No expressions in this context.
- Tree:** Width (px) [] Height (px) []
- memory:** Start address [] End address [] 8 [] No memory to display.
- breakpoints:** A checkbox for "main thread groups: it" is checked, with a condition of "1 hit". The hit address is `/home/roymanon/Desktop/OS-1/Operating-Systems-1/divide_by_zero.c:5 raise(SIGFPE);`.

Bottom Status Bar:

- gdbgui output (read-only)
- Copy/Paste available in all terminals with **ctrl+shift+v**
- Started new gdb process, pid 10430
- The program is not being run.
- Program output -- Programs being debugged are connected to this terminal. You can read output and send input to the program from here.

Terminal Log:

```
For help, type "help".
Type "apropos word" to search for commands related
to "word".
New UI allocated
(gdb) [Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/aarch64-linux
-gnu/libthread_db.so.1".

Breakpoint 1, main () at divide_by_zero.c:5
$ raise(SIGFPE);
```

Without debug flag:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ gcc -o divide_by_zero_no_debug divide_by_zero.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ ./divide_by_zero_no_debug
Floating point exception (core dumped)
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$
```

```
(gdb) where
#0  __pthread_killImplementation (threadid=261986772328128,
    signo=signo@entry=8, no_tid=no_tid@entry=0) at ./nptl/pthread_kill.c:44
#1  0x0000ee468c685444 in __pthread_kill_internal (signo=8,
    threadid=<optimized out>) at ./nptl/pthread_kill.c:78
#2  0x0000ee468c63b6ac in __GI_raise (sig=8) at ../sysdeps/posix/raise.c:26
#3  0x0000b0923a020764 in main ()
(gdb) █
```

```
(gdb) print var
$1 = var
(gdb)
```

3. invalid memory address

```
c invalid_memory.c > ...
1 #include <stdio.h>
2
3 int main() {
4     int *ptr = (int *)0xdeadbeef;
5     *ptr = 42;
6     return 0;
7 }
```

With debug flag:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ gcc -g -o invalid_memory invalid_memory.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ ./invalid_memory
Segmentation fault (core dumped)
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$
```

```
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000af3aa469072c in main () at invalid_memory.c:5
5          *ptr = 42;
(gdb) where
#0  0x0000af3aa469072c in main () at invalid_memory.c:5
(gdb) print var
$1 = var
(gdb)
```

The screenshot shows the gdbgui interface running in a browser window. The assembly code pane displays the main function with a breakpoint at line 5. The memory dump pane shows a variable \$1 = var. The threads pane shows a single thread stopped at the breakpoint. The breakpoints pane has a checked entry for main.

```
#!/usr/bin/gdb
#include <stdio.h>
int main() {
    int *ptr = (int *)0xdeadbeef;
    *ptr = 42;
    return 0;
}

(gdb) b main
Breakpoint 1, main () at invalid_memory.c:4
4          int *ptr = (int *)0xdeadbeef;
```

Program output -- Programs being debugged are connected to this terminal. You can read output and send input to the program from here.

Without debug flag:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ gcc -o invalid_memory_no_debug  
invalid_memory.c  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ ./invalid_memory_no_debug  
Segmentation fault (core dumped)  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$
```

```
Core was generated by `./invalid_memory_no_debug'.  
Program terminated with signal SIGSEGV, Segmentation fault.  
#0  0x0000c41593a6072c in main ()  
(gdb) where  
#0  0x0000c41593a6072c in main ()  
(gdb) print var  
$1 = var  
(gdb)
```

Makefile:

```
M Makefile
1 CC = gcc
2 CFLAGS = -Wall -g
3
4 TARGETS = stack_overflow divide_by_zero invalid_memory
5
6 all: $(TARGETS)
7
8 stack_overflow: stack_overflow.c
9 |     $(CC) $(CFLAGS) -o $@ $<
10 |
11 divide_by_zero: divide_by_zero.c
12 |     $(CC) $(CFLAGS) -o $@ $<
13 |
14 invalid_memory: invalid_memory.c
15 |     $(CC) $(CFLAGS) -o $@ $<
16 |
17 clean:
18 |     rm -f $(TARGETS) core* *.o
```

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ make
gcc -Wall -g -o stack_overflow stack_overflow.c
stack_overflow.c: In function ‘crash’:
stack_overflow.c:3:6: warning: infinite recursion detected [-Winfinite-recursion]
]
3 | void crash() {
| ^
stack_overflow.c:4:5: note: recursive call
4 |     crash();
| ^
gcc -Wall -g -o divide_by_zero divide_by_zero.c
gcc -Wall -g -o invalid_memory invalid_memory.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$ make clean
rm -f stack_overflow divide_by_zero invalid_memory core* *.o
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1$
```

תרגיל 2 - שימוש בספריה (נלמד בעזרת ספריית המרוכבים) - 10 נקודות

כתבו תוכנית הבודקת האם מספר מרוכב שייך [לכזאת מנделברוט](#) ([The Mandelbrot set](#)), על התוכניתן לקל 2 ארגומנטים המכילים את חלקי המספר (ה ממשי והמדומה) ורגומנט אופציונלי N (עם ערך ברירת מחדל לבחירתכם).

שיטת הבדיקה תהיה להלן: חשבו את $(c)_N^a$ (לפי ההגדירה בעמוד הוויקיפדיה בעברית), החליטו האם נראה שהסדרה מתבדרת ע"י השוואת של הערך המוחלט שלו (בעזרת `carg(carg)`) למספר גדול כלשהו M (קבוע לבחירתכם). הדפיסו הודעה בהתאם.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <complex.h>
4 #include <math.h>
5
6 int main(int argc, char *argv[]) {
7     if (argc < 3 || argc > 4) {
8         fprintf(stderr, "Usage: %s <real> <imag> [max_iterations](optional)\n", argv[0]);
9         return 1;
10    }
11    //atof - converts string (users input) to double
12    double real = atof(argv[1]);
13    double imag = atof(argv[2]);
14    //4 because argv[0] contains the programs name
15    int max_iter;
16    if (argc == 4) {
17        max_iter = atoi(argv[3]);
18    } else {
19        max_iter = 1000;
20    }
21
22    complex double c = real + imag * I;
23    complex double z = 0;
24    int n = 0;
25
26    while (n < max_iter && cabs(z) <= 2.0) {
27        z = z*z + c;
28        n++;
29    }
30
31    if (cabs(z) <= 2.0) {
32        printf("%.3f + %.3fi is in the Mandelbrot set\n", real, imag);
33    } else {
34        printf("%.3f + %.3fi is not in the Mandelbrot set\n", real, imag);
35    }
36
37    return 0;
38 }
```

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_2$ ./mandelbrot -1.627 -0.0
01
-1.627 + -0.001i is in the Mandelbrot set
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_2$ ./mandelbrot -1.629 -0.0
01
-1.629 + -0.001i is not in the Mandelbrot set
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_2$ ./mandelbrot -1.629 -0.0
01 5
-1.629 + -0.001i is in the Mandelbrot set
```

Invalid amount of arguments:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_2$ ./mandelbrot -1.629
Usage: ./mandelbrot <real> <imag> [max_iterations](optional)
```

Makefile:

```
M Makefile
1 CC = gcc
2 CFLAGS = -Wall -g
3 TARGET = mandelbrot
4 SRC = mandelbrot.c
5
6 all: $(TARGET)
7
8 $(TARGET): $(SRC)
9 |     $(CC) $(CFLAGS) -o $(TARGET) $(SRC) -lm
10 |
11 clean:
12 |     rm -f $(TARGET) core* *.o
```

תרגיל 3 - בניית ספריה - 10 נקודות

כמפורט לתרגיל 2 בנו את הפונקציה `is_in_mandelbrot` שחתימתה:

```
bool is_in_mandelbrot(complex double c, int N);
```

הבודקת האם מספר מרוכב שייר לקובץ מנדלבוט.

קמפלו אותה לספריה דינמית (כלומר shared object) בשם `libmandelbrot.so`. כתבו קובץ `header.h` המכוון על הפונקציה.

כתבו תוכנית המשמשת בספריה ("include" של ה-`header.h` וlienkg' עם ה-`shared object`) ובודקת שיכות עברו מספרים המתאימים בקלט מהמשתמש ע"י השורה:

```
scanf("%lf %lf", &real, &imag);
```

ומדפסה הודעה בהתאם. על התוכנית לעמוד כשהשתמש מכניס את המספר `0+0i` (שכמובן בקובוצה).

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_3$ make
gcc -Wall -fPIC -g -c mandelbrot.c
gcc -shared -o libmandelbrot.so mandelbrot.o
gcc -Wall -fPIC -g -o mandelbrot_program main.c -L. -lmandelbrot -lm
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_3$ ls
libmandelbrot.so  Makefile      mandelbrot.h  mandelbrot_program
main.c            mandelbrot.c  mandelbrot.o
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_3$ ./mandelbrot_program
Enter a complex number (real imag): 2 2
2.000 + 2.000i is not in the Mandelbrot set
Enter a complex number (real imag): -1.627 -0.001
-1.627 + -0.001i is in the Mandelbrot set
Enter a complex number (real imag): -1.629 -0.001
-1.629 + -0.001i is not in the Mandelbrot set
Enter a complex number (real imag): 0 0
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_3$
```

תרגיל 4 - code coverage - משקל 15 נקודות

באתר <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7> תוכל למצא מימוש (עובד) של אלגוריתם דייקסטרה. (תוכן לבחור ב-C או C++)

שנו את התוכנית כך שהתוכנית שלכם תתמוך (בתוך לולאות ((); for בקבלה גרפ' חדש, (קריית הגרף תבוצע מ-`scanf` בעזרת `stdin` או `cin` לבחירתכם), וודאו שקיים בדיקות תקיןות לקלט (כלומר לא שמי' יותר מדי או פחות מדי מרוחקים בשורה. דיקסטרה לא תומך במשקל'י קשותות שלילים) והרצת האלגוריתם.

הרצאה של מטריצה תקינה כמו בדוגמא ב-geeksforgeeks

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$ make
gcc -Wall -g -fprofile-arcs -ftest-coverage -o dijkstra dijkstra.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$ ./dijkstra
enter a 9x9 adjacency matrix (0 to quit):
0 4 0 0 0 0 0 8 0
4 0 8 0 0 0 0 11 0
0 8 0 7 0 4 0 0 2
0 0 7 0 9 14 0 0 0
0 0 0 9 0 10 0 0 0
0 0 4 14 10 0 2 0 0
0 0 0 0 0 2 0 1 6
8 11 0 0 0 0 1 0 7
0 0 2 0 0 0 6 7 0
Vertex      Distance from Source
0              0
1              4
2              12
3              19
4              21
5              11
6              9
7              8
8              14
```

יותר מדי מרחוקים בשורה:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$ ./dijkstra
enter a 9x9 adjacency matrix (0 to quit):
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 1
too many values in row 3.
invalid input, exiting.
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$
```

פחות מדי מרחוקים בשורה:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$ ./dijkstra
enter a 9x9 adjacency matrix (0 to quit):
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8
too few values in row 3.
invalid input, exiting.
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$
```

משקל שלילי:

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$ ./dijkstra
enter a 9x9 adjacency matrix (0 to quit):
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 -5 6 7 8 9
invalid input: negative edge detected
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$
```

gcov:

```
invalid input: negative edge detected
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$ gcov dijkstra.c
File 'dijkstra.c'
Lines executed:100.00% of 57
Creating 'dijkstra.c.gcov'

Lines executed:100.00% of 57
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_4$
```

תרגיל 5 - profiling - משקל 15 נקודות

מששו את שלושת הפתרונות לבעיית max sub array sum בסיבוכיות זמן ריצה של n^3 , n^2 , n .

התוכניות שלכם יקבלו שני ארגומנטים:
אחד - random seed (לשימוש עם srand)
שני - גודל הקלט (כמה המספרים שהתוכנית תיציר)

הקלט לשולשת האלגוריתמים יחולל באקראי (בעזרת קראיות ל(3) rand) בפונקציה "יעודית ורצת"
האלגוריתם תבצע גם היא בפונקציה. המספרים האקראים יכולים להתפלג בהצלחות אחידה בקטע (-25,

.74).
(אם תרצו התפלגות אחידה אחרת שימו לב שמספרים שליליים חייבים להככל אחרית התת קטע השלים יהיה
התת קטע המלא)

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$ make
gcc -Wall -g -pg -o max_subarray max_subarray.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$ ./max_subarray 42 100
results:
0(n^3) max sum: 2075
0(n^2) max sum: 2075
0(n)   max sum: 2075

timing (CPU cycles):
generation: 0.00 sec
0(n^3):    0.00 sec
0(n^2):    0.00 sec
0(n):      0.00 sec
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$ ./max_subarray 42 1000
results:
0(n^3) max sum: 25530
0(n^2) max sum: 25530
0(n)   max sum: 25530

timing (CPU cycles):
generation: 0.00 sec
0(n^3):    0.39 sec
0(n^2):    0.00 sec
0(n):      0.00 sec
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$ ./max_subarray 42 10000
[
```

$(10,000)^3$ takes too long to compute..

We will omit the n^3 once to see n, n^2 run on 10,000

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$ ./max_subarray 42 10000
results:
0(n^2) max sum: 247358
0(n)   max sum: 247358

timing (CPU cycles):
generation: 0.00 sec
0(n^2):     0.13 sec
0(n):       0.00 sec
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$ █
```

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$ gprof max_subarray gmon.out > analysis.txt
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$ ls analysis.txt gmon.out Makefile max_subarray max_subarray.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_5$
```

Flat profile:

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
97.44	0.38	0.38	1	380.00	380.00	max_subarray_n3
2.56	0.39	0.01	1	10.00	10.00	max_subarray_n2
0.00	0.39	0.00	1	0.00	0.00	generate_array
0.00	0.39	0.00	1	0.00	0.00	max_subarray_n

תרגיל 6 - תקשורת בעזרת סיגנלים - 20 נקודות

בתרגיל זה נכתבו 2 תוכניות שמתקשנות בעזרת סיגנלים - .signal_sender, signal_receiver על התוכניות להיות מסוגלות לשЛОח ולקבל מספר בן 8 ביטים. התקשרות תעשה בעזרת הסיגנלים SIGUSR1 ו-SIGUSR2, כאשר כל אחד מייצג ערך שהbeit יכול לקבל. השולח יבקש מהמשתמש את ה-PID של המקלט ומספר ישלח את המספר בעזרת הסיגנלים. המקלט יקלוט את הסיגנלים ובעזרת מטפל/מטפליים "יעוד"ים ירשום את הביטים ויבנה את המספר (בעזרת שימוש במשתנים גלובליים) ובסוף ידפיס את המספר למשתמש.

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_6$ make
gcc -Wall -g -o signal_sender signal_sender.c
gcc -Wall -g -o signal_receiver signal_receiver.c
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_6$ ./signal_receiver
My PID is 3703
Received 42
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_6$ 
```

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_6$ ./signal_sender
Enter receiver PID: 3703
Enter message (0-255): 42
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_6$ 
```

What happens if a signal is received during sleep and what should be done after?

When a signal is received during sleep, the sleep is interrupted, it doesn't resume. The handler is executed and after it finished the process continues from where it was after the sleep.

What should be done?

Use pause instead of sleep

Or,

Use a loop around sleep that makes sure the sleep continues from when it was interrupted.

תרגיל 7 - שימוש ב-pipes, יצירת תהליכיים - 20 נקודות.

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ echo "Avner Harishon,03-1234567" | ./add2PB  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ echo "Avner Hasheni,050-9876543" | ./add2PB  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ echo "ploni almoni,054-551634" | ./add2PB  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ echo "mom,052-5551634" | ./add2PB  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ echo "Bat Sheva Cohen,052-5552534" | ./add2PB  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$
```

```
C add2PB.c      ≡ phones.txt x  
≡ phones.txt  
1 Avner Harishon,03-1234567  
2 Avner Hasheni,050-9876543  
3 ploni almoni,054-5551634  
4 mom,052-5551634  
5 Bat Sheva Cohen,052-5552534  
6
```

```
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ ./findPhone "Avner Harishon"  
03-1234567  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ ./findPhone "Avner Hasheni"  
050-9876543  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ ./findPhone "ploni almoni"  
054-5551634  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ ./findPhone "mom"  
052-5551634  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$ ./findPhone "Bat Sheva Cohen"  
052-5552534  
roymamon@roy:~/Desktop/OS-1/Operating-Systems-1/Part_7$
```

