

```
● shayg@ShayG:~/os_course/assignments/ex1$ ls
makefile q1 q2 q3 q4 q5 q6
```

כדי לקרוא ל `make` של כל אחת מהתיקיות השתמשנו בדגל `-C` כדי שהוא יעשה `cd` לתיקייה ויריץ שם `make all`

```
5  .PHONY: all clean q1 q2 q3 q4 q5 q6
6
7  all: q1 q2 q3 q4 q5 q6
8
9  # run make in each subdirectory
10 q*:
11     make -C $@ all
12
13 clean:
14     make -C q1 clean
15     make -C q2 clean
16     make -C q3 clean
17     make -C q4 clean
18     make -C q5 clean
19     make -C q6 clean
```

שאלה 1

כדי שנוכל לקבל את coren הרצנו את הפקודה:

```
ulimit -c unlimited
```

בעצם נתנו למערכת הפעלה ליצור קבצי core בגודל בלתי מוגבל

תוכנה 1:

גלישה מהמחשית

```
5 void stack_overflow() { stack_overflow(); }
6
7 int main(void) {
8     stack_overflow();
9     return 0;
10 }
```

הרצנו את התוכנה הראשונה (שגומרת ל stack over flow) כאשר בקימפול לא שלחנו את הדגל -g

```
shayg@ShayG:~/os_course/assignments/ex1/q1$ ./bad_prog_1
Segmentation fault (core dumped)
```

ונוצר לנו הקובץ הבא:

```
-rw----- 1 shayg shayg 8499200 Apr 15 19:22 core
```

עכשיו נפתח את coren

```
shayg@ShayG:~/os_course/assignments/ex1/q1$ gdb ./bad_prog_1 core
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./bad_prog_1...
(No debugging symbols found in ./bad_prog_1)
[New LWP 25945]
Core was generated by `./bad_prog_1'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x000055af6e3a0136 in stack_overflow ()
(gdb)
```

כאשר הרצנו את הקובץ השני (עם הדגל g- שנותן לנו מידע עבור דיאבג)

```
shayg@ShayG:~/os_course/assignments/ex1/q1$ ./bad_prog_1_g
Segmentation fault (core dumped)
shayg@ShayG:~/os_course/assignments/ex1/q1$ gdb ./bad_prog_1_g core
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./bad_prog_1_g...
[New LWP 26970]
Core was generated by `./bad_prog_1_g'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x000055da6242f136 in stack_overflow () at bad_prog_1.c:5
5      void stack_overflow() { stack_overflow(); }
(gdb)
```

ההבדל העיקרי הוא בעצם השורה האחרונה (וגם בסוף השורה לפניה), שמדפיסה לנו את השורה שממנה השגיאה נזרקה.
כאשר נדפיס את מחסנית הקריאות עם הפקודה where נקבל:

```
(gdb) where
#0  0x000055da6242f136 in stack_overflow () at bad_prog_1.c:5
#1  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#2  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#3  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#4  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#5  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#6  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#7  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#8  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#9  0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#10 0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#11 0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#12 0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#13 0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#14 0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
#15 0x000055da6242f13b in stack_overflow () at bad_prog_1.c:5
```

זה ממשיך עוד אחרי (בסוף המחסנית מלאה) אז צילמתי רק את ההתחלה
אין משתנים אז אין טעם להשתמש בprint.

```

6   int main(void) {
7       int x = 1;
8       int y = 0;
9       int z = x / y;
10      printf("%d\n", z);
11      return 0;
12  }

```

התוכנית כאשר קמפלנו אותה ללא דגל g:

```

shayg@ShayG:~/os_course/assignments/ex1/q1$ ./bad_prog_2
Floating point exception (core dumped)
shayg@ShayG:~/os_course/assignments/ex1/q1$ gdb ./bad_prog_2 core
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./bad_prog_2...
(No debugging symbols found in ./bad_prog_2)
[New LWP 31476]
Core was generated by `./bad_prog_2'.
Program terminated with signal SIGFPE, Arithmetic exception.
#0  0x0000561a5c1ba167 in main ()

```

כשהרצתי את התוכנית שהיא מקומפלט עם g-

```

shayg@ShayG:~/os_course/assignments/ex1/q1$ ./bad_prog_2_g
Floating point exception (core dumped)
shayg@ShayG:~/os_course/assignments/ex1/q1$ gdb ./bad_prog_2_g core
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./bad_prog_2_g...
[New LWP 1053]
Core was generated by `./bad_prog_2_g'.
Program terminated with signal SIGFPE, Arithmetic exception.
#0  0x000055df7a0f4167 in main () at bad_prog_2.c:9
9      int z = x / y;

```

ניתן לראות שהתוכנית נפלה בשורה 9 בגלל בעיה אריתמטית.

נדפיס את המשתנים כדי לראות מה קרה:

```
(gdb) where
#0  0x000055df7a0f4167 in main () at bad_prog_2.c:9
(gdb) print x
$1 = 1
(gdb) print z
$2 = 0
(gdb) print y
$3 = 0
(gdb) 
```

ואפשר לראות שחילקנו ב0.

הגדרנו מצביע ל NULL (הוא יצביע לכתובת 0), ואי אפשר לגשת לכתובת הזאת.

```

7   int main(void) {
8       int *p = NULL; // p is a null pointer
9       *p = 42;
10      return 0;
11  }
```

כאשר נריץ את gdb כשדיבגנו בלי דגל הג נקבל:

```

shayg@ShayG:~/os_course/assignments/ex1/q1$ ./bad_prog_3
Segmentation fault (core dumped)
shayg@ShayG:~/os_course/assignments/ex1/q1$ gdb ./bad_prog_3 core
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./bad_prog_3...
(No debugging symbols found in ./bad_prog_3)
[New LWP 4207]
Core was generated by `./bad_prog_3'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x00005612e880113d in main ()
(gdb) where
#0  0x00005612e880113d in main ()
(gdb) print p
No symbol "p" in current context.
```

```
shayg@ShayG:~/os_course/assignments/ex1/q1$ ./bad_prog_3_g
Segmentation fault (core dumped)
shayg@ShayG:~/os_course/assignments/ex1/q1$ gdb ./bad_prog_3_g core
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

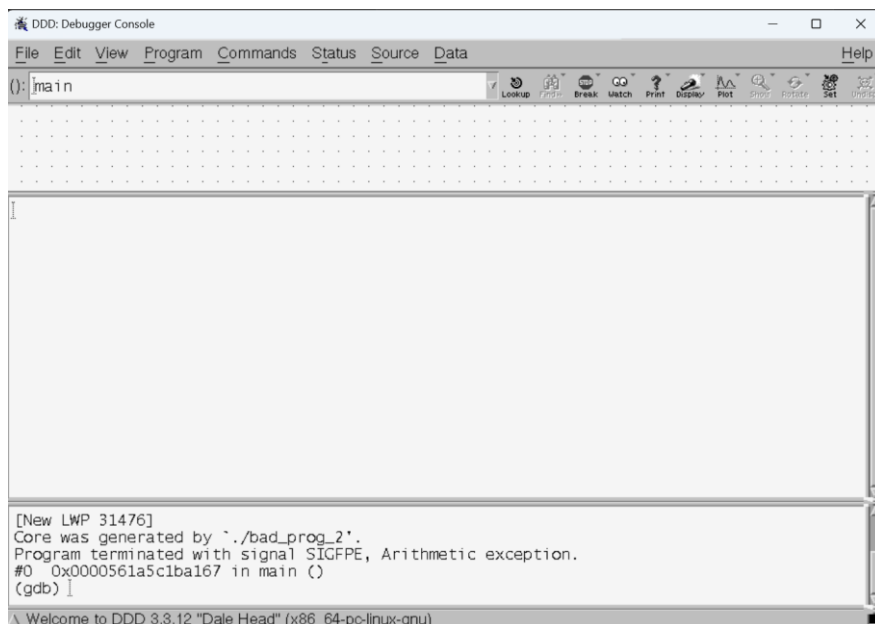
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./bad_prog_3_g...
[New LWP 4461]
Core was generated by `./bad_prog_3_g'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x0000559a8ea4613d in main () at bad_prog_3.c:9
9      *p = 42;
(gdb) where
#0  0x0000559a8ea4613d in main () at bad_prog_3.c:9
(gdb) print p
$1 = (int *) 0x0
```

לאחר מכן הדפסתי את המשתנה שלנו (המצביע p) ואפשר לראות שהסוג שלו הוא מצביע ל int והערך שלו הוא 0.

DDD

כדי לדאבג עם את coren עם ddd הרצנו את הפקודה: `ddd -core=core ./bad_prog1`
נשים לב שהתוכנית מקומפלת עם דגל `g` שנוכל לדאבג.

אם לא היינו מקמפלים עם הדגל היינו מקבלים מסך ריק (בלי הקוד), למטה את השגיאה:



תוכנית 1

(אין מה לדאבג כאן יותר מידי, התוכנית די פשוטה)



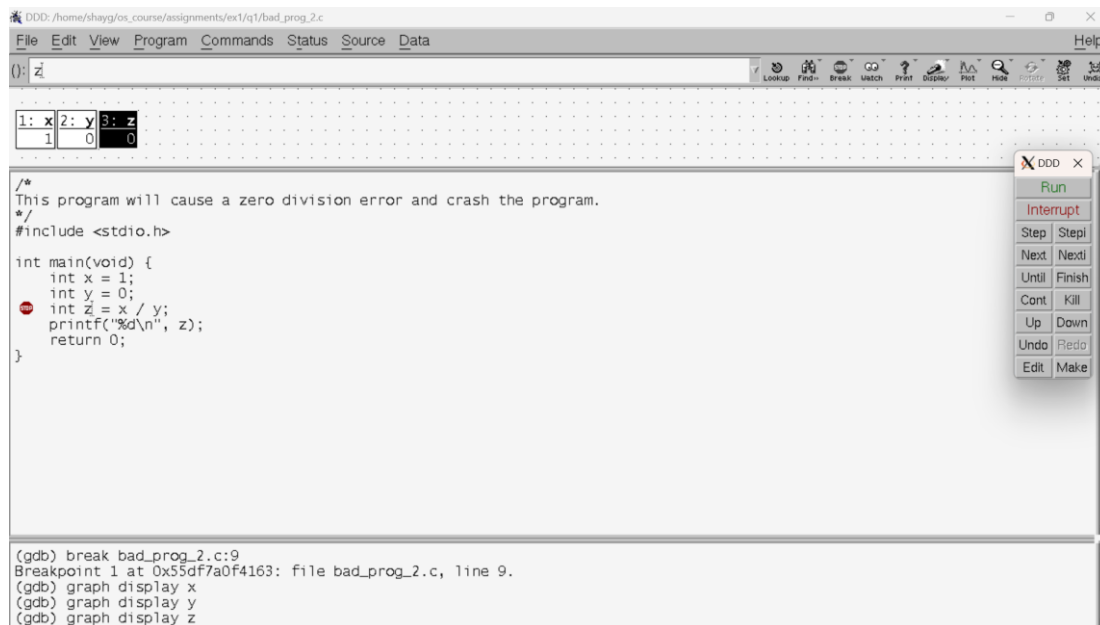
תוכנית 2

הפעם יש באמת מה לדאוג,

שמתי break point בשורה שזה אמור ליפול, והצגתי למעלה את המשתנים שלנו (מסומן בצהוב)

הרצתי את התוכנית והיא עכשיו עצרה

כשלחצתי על step הוא ביצע את השורה, וזרק את השגיאה



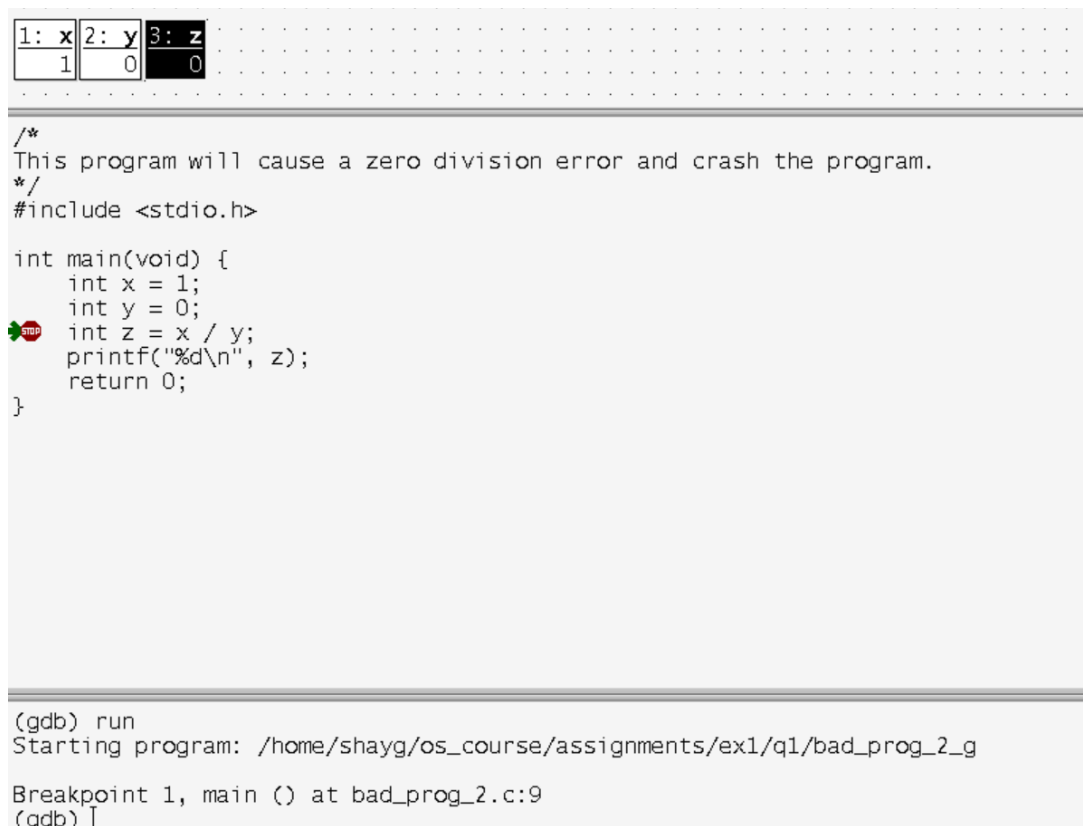
```
/*
This program will cause a zero division error and crash the program.
*/
#include <stdio.h>

int main(void) {
    int x = 1;
    int y = 0;
    int z = x / y;
    printf("%d\n", z);
    return 0;
}
```

(gdb) break bad_prog_2.c:9
Breakpoint 1 at 0x55df7a0f4163: file bad_prog_2.c, line 9.
(gdb) graph display x
(gdb) graph display y
(gdb) graph display z

לאחר מכן חלצתי על run והוא הגיע ב break point ששמתי (מסומן בכחול)

אפשר לראות את החץ הירוק שמצביע על איזה שורה אנחנו נמצאים



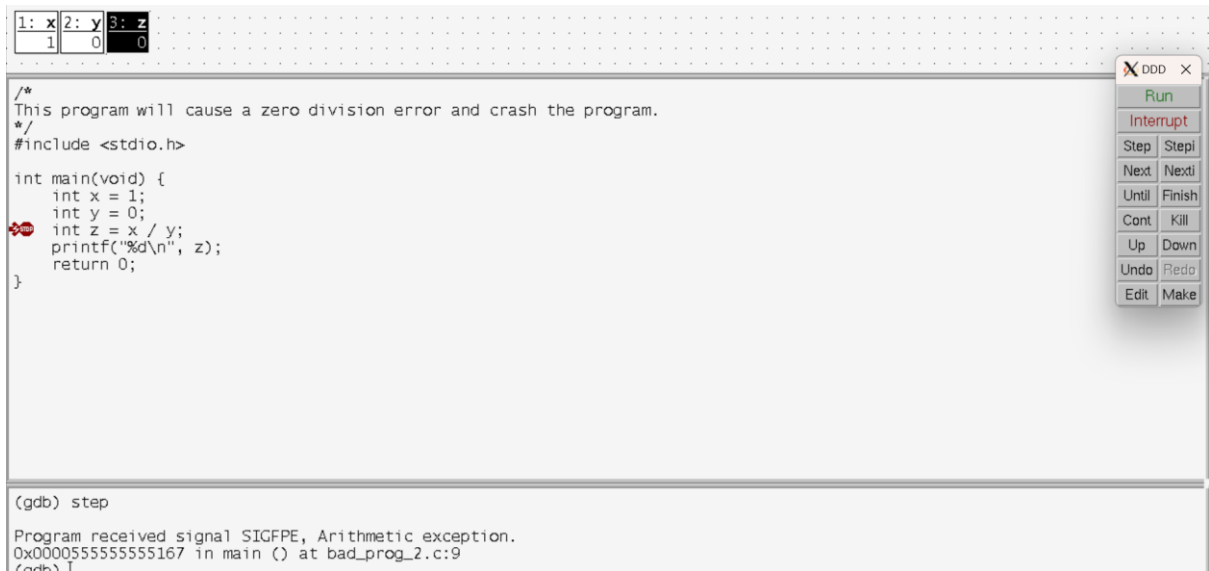
```
/*
This program will cause a zero division error and crash the program.
*/
#include <stdio.h>

int main(void) {
    int x = 1;
    int y = 0;
    int z = x / y;
    printf("%d\n", z);
    return 0;
}
```

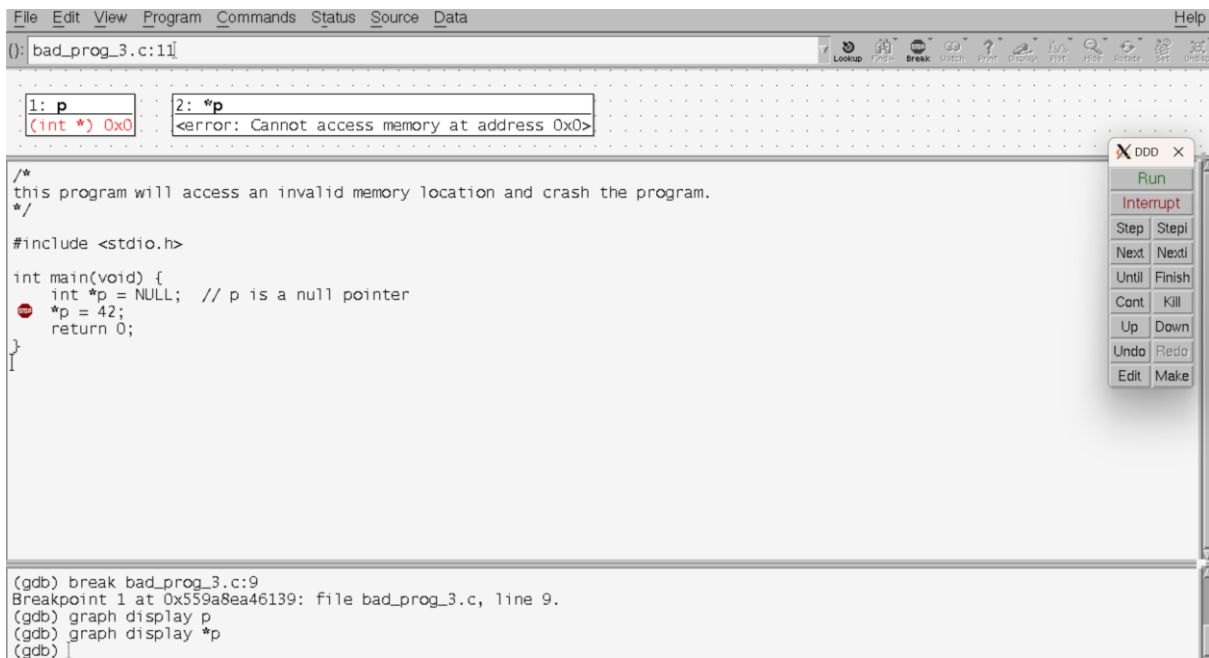
(gdb) run
Starting program: /home/shayg/os_course/assignments/ex1/q1/bad_prog_2_g
Breakpoint 1, main () at bad_prog_2.c:9
(gdb) |

כשלחצתי על step הוא ביצע את השורה והוא נפל.

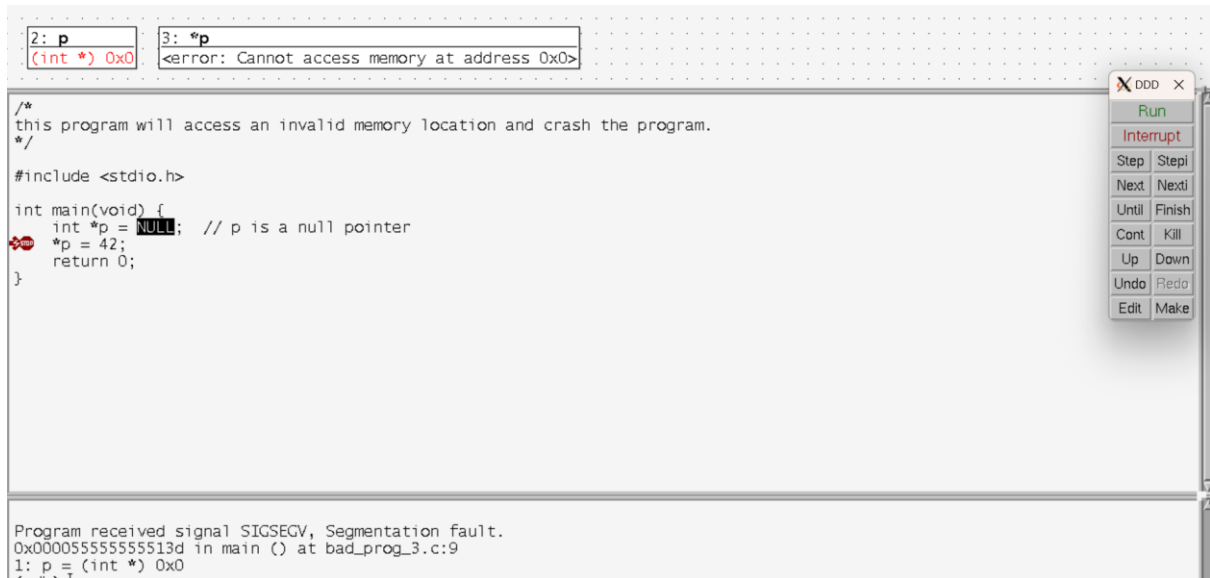
אפשר לראות את החץ בצבע אדום שמסמן את הנפילה.



תוכנית 3:

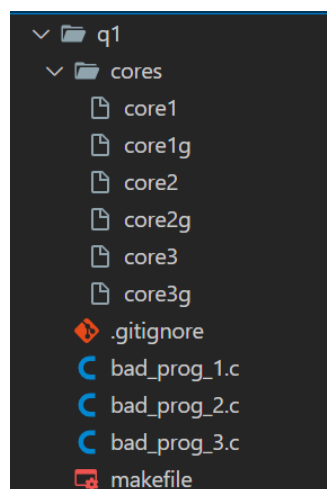


גם כאן הרצתי, הוא נכנס לbreak point וכשהמשכתי את הריצה הוא נפל, וddd מסמן לי איפה הוא נפל.



המבנה של הקבצים הוא:

בתיקייה הראשית יש את שלושת התוכניות, makefile וקובץ .gitignore. שעוזר לי לעבוד עם git, ועוד תיקייה שבה יש את כל coren שנוצרו בריצה של התוכניות. קובץ שנגמר בכ מסמן שהתוכנית קומפלה עם הדגל -g.



הקובץ make נראה ככה:

```
q1 > makefile
1  .PHONY: all clean
2
3  CC = gcc
4
5  PROGS = bad_prog_1 bad_prog_2 bad_prog_3 bad_prog_1_g bad_prog_2_g bad_prog_3_g
6
7  all: $(PROGS)
8
9
10 # for cores with debugging information (with -g option)
11 bad_prog_%_g: bad_prog_%.c
12     $(CC) -c -Wall -g -o $@.o $<
13     $(CC) -o $@ $@.o
14
15
16 # for cores without debugging information
17 bad_prog_%: bad_prog_%.c
18     $(CC) -c -Wall -o $@.o $<
19     $(CC) -o $@ $@.o
20
21
22 clean:
23     rm -f *.o $(PROGS) core
```

הוא יוצר 6 תוכניות – לכל אחת משלושת התוכניות שכתבנו הוא יוצר 2, אחת מקומפלט עם הדגל g ואחת לא.

הפונקציה שמחשבת את ההסתברות:

```
18 long double poisson(int k, double lambda) {
19     return (pow(lambda, k) / factorial(k)) * exp(-lambda);
20 }
```

בעזרת הפונקציה `exp` אחנו משיגים דיוק של `long double`

פונקציית ה-`main`:

אנחנו בודקים שקיבלנו ביוק 3 פרמטרים, ואז בודקים את תקינות הקלט, לאחר מכן מחשבים את ההסתברות ומדפיסים.

```
22 int main(int argc, char const *argv[]) {
23     // check for correct number of arguments
24     if (argc != 3) {
25         printf("Usage: ./Poisson <lambda> <k>\n");
26         return 1;
27     }
28     // get lambda and k from command line arguments
29     double lambda;
30     int k;
31     if (sscanf(argv[1], "%lf", &lambda) != 1) {
32         printf("λ need to be a number\n");
33         return 1;
34     }
35
36     if (sscanf(argv[2], "%d", &k) != 1) {
37         printf("k need to be an integer\n");
38         return 1;
39     }
40
41     long double val = poisson(k, lambda);
42     printf("P_X(%d) = %.10Lf\n", k, val);
43     return 0;
44 }
```

הרצת התוכנית:

- הרצה של `make`
- הרצה ללא ארגומנטים
- הרצה עם ארגומנטים שונים

```

shayg@ShayG:~/os_course/assignments/ex1/q2$ make
gcc -c -Wall -o Poisson.o Poisson.c
gcc -o Poisson Poisson.o -lm
shayg@ShayG:~/os_course/assignments/ex1/q2$ ./Poisson
Usage: ./Poisson <lambda> <k>
shayg@ShayG:~/os_course/assignments/ex1/q2$ ./Poisson 2 1
P_X(1) = 0.2706705665
shayg@ShayG:~/os_course/assignments/ex1/q2$ ./Poisson 2 10
P_X(10) = 0.0000381899
shayg@ShayG:~/os_course/assignments/ex1/q2$ ./Poisson 2 2
P_X(2) = 0.2706705665
shayg@ShayG:~/os_course/assignments/ex1/q2$ ./Poisson 3 3
P_X(3) = 0.2240418077
shayg@ShayG:~/os_course/assignments/ex1/q2$ ./Poisson 3 5
P_X(5) = 0.1008188134

```

הקימפול והלינקוג' עם הדלג lm

```

7 Poisson: Poisson.c
8     $(CC) -c -Wall -o $@.o $<
9     $(CC) -o $@ $@.o -lm
10

```

שאלה 3

הקבצים שיש לנו:

```
shayg@ShayG:~/os_course/assignments/ex1/q3$ ls
main.c  makefile  poisson.c  poisson.h
```

נשתמש ב `poisson.c` כדי ליצור את הספרייה, וב `poisson.h` בקובץ `main.c` כדי להשתמש בה.

הקובץ `make`:

```
1 .PHONY: all clean
2
3 CC = gcc
4
5 all: main libpoisson.so
6
7 main: main.c libpoisson.so
8     $(CC) -o main main.c -L. -lpoisson -lm
9
10 libpoisson.so: poisson.o
11     $(CC) -shared -o libpoisson.so poisson.o
12
13 poisson.o: poisson.c poisson.h
14     $(CC) -c -fPIC poisson.c
15
16 clean:
17     rm -f main libpoisson.so poisson.o
```

כדי ליצור את הספרייה נשתמש בדגל `-shared` ואת לקובץ שיווצר נקרא `libpoisson.so`.

בלינקוג' נשתמש ב `-L.` בשביל להגיד ללינקר להסתכל בתיקייה הנוכחית, ב `-lpoisson` שזה השם של הספרייה ללא ההתחלה של `lib` ולא הסיומת של `.so`, ובדגל `-lm` כדי להשתמש בספרייה `math`.

בנוסף נצטרך להגיד ל `dynamic linker` לחפש סיפריות דינמיות גם בתיקייה הזאת לכן חשוב להריץ את הפקודה

`export LD_LIBRARY_PATH=.`

כדי שהוא יידע לקחת את הספרייה מהתיקייה הנוכחית.

כשנקמפל את הקובץ `poisson.c` נשתמש בדגל `-fPIC` שזה אומר `Position Independent Code` זה בשביל שנוכל להשתמש בו בספרייה דינאמית, וזה אומר שהקובץ שהמקופל לא יהיה תלוי במיקום של הקובץ בשביל לעבוד.

הרצה:

```
shayg@ShayG:~/os_course/assignments/ex1/q3$ make
gcc -c -fPIC poisson.c
gcc -shared -o libpoisson.so poisson.o
gcc -o main main.c -L. -lpoisson -lm
shayg@ShayG:~/os_course/assignments/ex1/q3$ ./main

~~~~~ Poisson distribution ~~~~~

  λ      |      k      |      P_X(k)
-----|-----|-----
      2   |      1   | 0.270670566473
      2   |     10   | 0.000038189851
      2   |      2   | 0.270670566473
      3   |      3   | 0.224041807655
     100   |      3   | 0.000000000000
```

שאלה 4:

הקוד שמופיע ב [geeksforgeeks](https://www.geeksforgeeks.org/) מניח שהגודל של הגרף קבוע ל 9 קודקודים, כדי להתגבר על זה אני קיבלתי כקלט את הגודל מהמשתמש, והקצתי את המטריצה בצורה דינמית.

כדי להגיע לכיסוי מלא, כתבתי 10 קבצי קלטים רעים, וקובץ קלט רע אחד, הם מופיעים בתיקייה `q4/inputs` ויש שם קובץ `inputs.txt` שמסביר מה כל קלט בודק.

הסבר:

- קלטים רעים:
 0. קלט ריק
 1. מספר שלילי של קודקודים
 2. לא שולחים מטריצה בכלל
 3. שולחים צלע שלילית
 4. שולחים צלע שהיא לא 0 לאלכסון (הגרף אמור להיות פשוט)
 5. לא שולחים מאיזה קודקוד להתחיל את האלגוריתם
 6. שולחים קודקוד התחלה שהוא לא בגרף
 7. שולחים יותר מידי קלט
 8. שולחים מספר גדול מידי של קודקודים שהוא לא יצליח להקצות את המטריצה
 9. שולחים מספר גדול מידי של קודקודים, אבל לא גדול מידי, שהוא יצליח להקצות את המטריצה אבל ייכשל להקצות את אחת מהשורות.
- קלט טוב: קלט שיעבוד.

:Makefile

כדי שנוכל לבדוק את הכיסוי נקמפל עם הדגלים `-fprofile-arcs -ftest-coverage` ובליקוג' נשתמש בדגל `-lgcov`

הכנתי גם target שיעזור לי להריץ את כל הקלטים שהכנתי, קראתי לו `run`.

הוא מריץ את הקלט הטוב, ואז הוא עובד על כל הקבצים הרעים ומריץ אותם אחד אחרי השני. את כל הפלט אני רושם ל `./dev/null`

בסוף אני מריט `main.c` gcov כדי לראות את אחוז הכיסוי.

```
1  .PHONY: all run clean
2
3  CC = gcc
4
5  all: main
6
7  main: main.c
8      $(CC) -fprofile-arcs -ftest-coverage -c -Wall -o $@.o $<
9      $(CC) -o $@ $@.o -lgcov
10
11 # for running the program with the good and bad inputs
12 # we clean the previous gcov files, compile the program, run the program with the bad inputs and the good input
13 run: main
14     ./main < inputs/good_input.txt > /dev/null
15     for file in ./inputs/bad_input*.txt; do \
16     ./main < $$file > /dev/null || true; \
17     done
18     gcov main.c
19
20 clean:
21     rm -f *.o main *.gcno *.gcda *.gcov
```


דוגמא של הרצה:

```
● shayg@ShayG:~/os_course/assignments/ex1/q4$ make run
gcc -fprofile-arcs -ftest-coverage -c -Wall -o main.o main.c
gcc -o main main.o -lgcov
./main < inputs/good_input.txt > /dev/null
for file in ./inputs/bad_input*.txt; do \
    ./main < $file > /dev/null || true; \
done
gcov main.c
File 'main.c'
Lines executed:100.00% of 82
Creating 'main.c.gcov'
```

תרגיל 5:

כדי להריץ עם התוכנית יש כמה דרכים:

1. להריץ כל תכונת אחת אחת עם הפרמטרים 100,1000,10000
 2. כתבתי target בשם run שגם מכין קבצי output שמריץ לפי משתנה N שאפשר להעביר כארגומנט לmake
- אז אפשר להריץ את התוכנית עם $n = 100$ ככה:

```
shayg@ShayG:~/os_course/assignments/ex1/q5$ make run N=100
./main1 5 100 >> /dev/null
echo "~ maximun subarray O(n) ~" > output100.txt
gprof --flat-profile main1 | head >> output100.txt
mv gmon.out gmon_outputs/gmon.out.main1_100
./main2 5 100 >> /dev/null
echo "\n\n~\n\n" >> output100.txt
echo "~ maximun subarray O(n^2) ~\n" >> output100.txt
gprof --flat-profile main2 | head >> output100.txt
mv gmon.out gmon_outputs/gmon.out.main2_100
./main3 5 100 >> /dev/null
echo "\n\n~\n\n" >> output100.txt
echo "~ maximun subarray O(n^3) ~\n" >> output100.txt
gprof --flat-profile main3 | head >> output100.txt
mv gmon.out gmon_outputs/gmon.out.main3_100
```

שמרתי את זה בקובץ בשם output\$(N).txt

הפקודה gprof --flat-profile main1 | head >> output\$(N).txt

תיצור הפלט של gprof (בתצורה יותר מתומצתת בעזרת --flat-profile), ותעביר את הפלט ל head שיביא רק את 10 השורות הראשונות, ונכתוב את זה לקובץ טסקט.

הפלט נראה ככה: (עם טיפה עריכה)

~maximun subarray O(n)~

Flat profile:

Each sample counts as 0.01 seconds.

no time accumulated

	%	cumulative	self		self	total	
time	seconds	seconds	calls	Ts/call	Ts/call	name	
0.00	0.00	1	0.00	0.00	0.00	generate_random_array	
0.00	0.00	1	0.00	0.00	0.00	max_sub_array	

~maximun subarray O(n^2)~

Flat profile:

Each sample counts as 0.01 seconds.

no time accumulated

	%	cumulative	self		self	total	
time	seconds	seconds	calls	Ts/call	Ts/call	name	
0.00	0.00	1	0.00	0.00	0.00	generate_random_array	

0.00 0.00 1 0.00 0.00 0.00 max_sub_array

~~~~~

~maximun subarray  $O(n^3)$ ~

Flat profile:

Each sample counts as 0.01 seconds.

no time accumulated

% cumulative self self total

time seconds seconds calls Ts/call Ts/call name

0.00 0.00 1 0.00 0.00 0.00 generate\_random\_array

0.00 0.00 1 0.00 0.00 0.00 max\_sub\_array

אפשר לראות שעבור קלט קטן כל הפונקציות עובדות בערך אותו דבר.

$n = 1000$

~ maximun subarray  $O(n)$  ~

Flat profile:

Each sample counts as 0.01 seconds.

no time accumulated

% cumulative self self total

time seconds seconds calls Ts/call Ts/call name

0.00 0.00 0.00 1 0.00 0.00 generate\_random\_array

0.00 0.00 0.00 1 0.00 0.00 max\_sub\_array

~~~~~

~ maximun subarray $O(n^2)$ ~

Flat profile:

Each sample counts as 0.01 seconds.

no time accumulated

% cumulative self self total

time seconds seconds calls Ts/call Ts/call name

0.00 0.00 0.00 1 0.00 0.00 generate_random_array

0.00 0.00 0.00 1 0.00 0.00 max_sub_array

~~~~~

~ maximun subarray  $O(n^3)$  ~

Flat profile:

Each sample counts as 0.01 seconds.

| %      | cumulative | self    | self  | total   |         |                       |  |
|--------|------------|---------|-------|---------|---------|-----------------------|--|
| time   | seconds    | seconds | calls | ms/call | ms/call | name                  |  |
| 101.10 | 0.54       | 0.54    | 1     | 535.82  | 535.82  | max_sub_array         |  |
| 0.00   | 0.54       | 0.00    | 1     | 0.00    | 0.00    | generate_random_array |  |

אפשר לראות שעבור  $n^2$  ו  $n$  קיבלנו תוצאה דומה ל  $n = 100$  ועבור  $n^3$  קיבלנו שהוא רץ ב 0.54 שניות, כאשר כמעט כל הזמן בוזבז על למצוא את התת מערך המקסימלי.

$n = 10000$

~ maximun subarray  $O(n)$  ~

Flat profile:

Each sample counts as 0.01 seconds.

no time accumulated

| %    | cumulative | self    | self  | total   |         |                       |  |
|------|------------|---------|-------|---------|---------|-----------------------|--|
| time | seconds    | seconds | calls | Ts/call | Ts/call | name                  |  |
| 0.00 | 0.00       | 0.00    | 1     | 0.00    | 0.00    | generate_random_array |  |
| 0.00 | 0.00       | 0.00    | 1     | 0.00    | 0.00    | max_sub_array         |  |

~~~~~  
~ maximun subarray $O(n^2)$ ~

Flat profile:

Each sample counts as 0.01 seconds.

%	cumulative	self	self	total			
time	seconds	seconds	calls	ms/call	ms/call	name	
101.12	0.18	0.18	1	182.01	182.01	max_sub_array	
0.00	0.18	0.00	1	0.00	0.00	generate_random_array	

~~~~~  
~ maximun subarray  $O(n^3)$  ~

Flat profile:

Each sample counts as 0.01 seconds.

| %      | cumulative | self    | self  | total  |        |                       |  |
|--------|------------|---------|-------|--------|--------|-----------------------|--|
| time   | seconds    | seconds | calls | s/call | s/call | name                  |  |
| 101.10 | 376.81     | 376.81  | 1     | 376.81 | 376.81 | max_sub_array         |  |
| 0.00   | 376.81     | 0.00    | 1     | 0.00   | 0.00   | generate_random_array |  |

אפשר לראות שעבור האלגוריתם שרץ ב  $O(n)$  התוצאות זהות למה שהיה קודם.

עבור  $O(n^2)$  קיבלנו שהוא רץ ב-0.18 שניות כאשר רוב הזמן היה על למצוא את התת מערך המקסימלי.

עבור  $O(n^3)$  קיבלנו שהוא רץ ב-376.81 שניות (!! כלומר הוא ממש לא יעיל.

כדי להוסיף אדם לספר טלפונים, אנחנו מקבלים את הערכים בתוך הארגומנטים, ומשתמשים ב `echo` כדי להכניס את המידע לקובץ.

אנחנו רצים בלולאה על הערכים שהתקבלו בארגומנטים, יוצרים תהליך בן חדש, שבו בעזרת הפקודה `echo` מכניסים את המידע לספר טלפונים.

הנחה חשובה – הקלט הוא תקין, כלומר תמיד יעבירו לנו קלט בצורה טובה

לפני ההרצה:

The screenshot shows a code editor with two tabs: `phonebook.txt` and `findPhone.c`. The `phonebook.txt` file contains two lines: `1 Shay G,052-1234556` and `2`. Below the editor, the `TERMINAL` tab shows a single command: `shayg@ShayG:~/os_course/assignments/ex1/q6$ ./add2PB itzik hoi,052-3334943`.

אחרי:

The screenshot shows the same code editor. The `phonebook.txt` file now contains three lines: `1 Shay G,052-1234556`, `2 itzik hoi,052-3334943`, and `3`. The `TERMINAL` tab shows two commands: `shayg@ShayG:~/os_course/assignments/ex1/q6$ ./add2PB itzik hoi,052-3334943` and `shayg@ShayG:~/os_course/assignments/ex1/q6$`.

כדי לחפש אנחנו ממשים את הפקודה הבאה:

```
grep <name> phonebook.txt | cut -d, -f 2 | sed 's/ //g'
```

ומשתמשים ב `fork` ו `execlp` כדי להריץ כל פעם פקודה חדשה וב `pipe` כדי להעביר מידע בין התהליכים האלו.

דוגמת הרצה:

PROBLEMS

7

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

- shayg@ShayG:~/os\_course/assignments/ex1/q6\$ ./findPhone itzik  
052-3334943
- shayg@ShayG:~/os\_course/assignments/ex1/q6\$ ./findPhone "itzik hoi"  
052-3334943
- shayg@ShayG:~/os\_course/assignments/ex1/q6\$ █