Laborator 05

Pentru aceste laborator vom folosi utilitarul **valgrind**. Pentru a instala valgrind:

```
sudo apt install valgrind
```

Valgrind vă va ajuta (nu mereu) să descoperiți bug-uri ce pot duce la erori de tip seg fault, stack smashing, abort, etc.. Îl puteți folosi din linia de comandă sub următoarele forme:

```
valgrind ./executabil
```

```
valgrind --tool=exp-sgcheck ./executabil
```

Pentru unele exerciții va fi utilă prima formă, pentru altele cea de-a doua.

Puteți ignora linia: `error calling PR_SET_PTRACER, vgdb might block`

Puteți ignora liniile: `warning: evaluate_Dwarf3_Expr: unhandled DW_OP_ 0x93`

Pentru fiecare exercițiu urmați pașii următori:

- Compilați (doar cu Debug, cu Launch unele erori nu vor apărea) și rulați.
- Notați în REPORT.txt cum s-a terminat programul, ce eroare a dat (posibil să execute corect, spuneți și asta).
- Analizați codul și notați în REPORT.txt descrierea bug-ului.
- Rulați cu valgrind și copiați în report DOAR liniile care identifică bug-ul.
- Dacă valgrind v-a ajutat să identificați bug-ul notați asta.
- Rezolvați bug-ul și rulați iar valgrind pentru a confirma.

Atenție este posibil să rulați și să vă rămână programul blocat deși de fapt este vorba de un seg fault sau abort. Dacă vă uitați la tab-ul Debug Console va scrie explicit Seg Fault:

```
Program received signal SIGSEGV, Segmentation fault.
```

Exemplu exercițiu REPORT.txt

```
EX 01_badFree.c:

Programul se termină cu Segmentation Fault.
Se apelează free peste un pointer care nu a fost alocat.

==1135== Invalid free() / delete / delete[] / realloc()
==1135==    at 0x483CA3F: free (in /usr/lib/x86_64-linux-
gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==1135==    by 0x10916F: main (1_badFree.c:7)
==1135==  Address 0x5 is not stack'd, malloc'd or (recently) free'd
```