# Valgrind as debugging tool

- Valgrind is also useful as a debugging tool, displaying information about potential errors detected during the run of a program
- Recompile the program using the -g flag first
- Run the program through valgrind as follows:

  valgrind -v ./yourprogram

- Valgrind will produce extra output before/during/after the program execution, identifying any potential problems it spots

# Valgrind for memory checking

- Valgrind can be used to detect memory leaks: cases where you allocate memory using malloc, calloc, new, etc but forget to appropriately deallocate it later (via free, delete, etc)
- To run with memory checking enabled, use the command

  valgrind -v –leak-check=full ./yourprogram
- A memory summary will be displayed after the program ends, broken down into a heap summary and leak summary

# Key types of leaks in summary

- “definitely lost”: probable leaks, the space was allocated but never deallocated
- “indirectly lost”: also probable leaks: typically because the memory that contained a pointer to an item was lost (e.g. you deallocated the root of a tree before deleting its children)

# Suggested order

- First fix all the basic errors found with

  valgrind -v ./yourprog
- Then test/fix with basic memory checking

  valgrind -v --leak-check=yes ./yourprog
- Finally, use full memory checking

  valgrind -v --leak-check=full ./yourprog