The University of Michigan
Electrical Engineering & Computer Science
EECS 281: Data Structures and Algorithms
Fall 2024

Lab 1: Tools for EECS 281

Instructions:
Work on this document with your group, then enter the answers on the canvas quiz.

26/26

Note:
Please **complete the Prelab** (found in the Lab 1 Canvas folder)! Be prepared before you meet with your lab group, and read this document so that you know what you must submit for full credit. You can even start it ahead of time and then ask questions during any lab section for help completing it.

You MUST include the following assignment identifier at the top of every file you submit to the autograder as a comment. This includes all source files, header files, and your Makefile (if there is one). If there is not autograder assignment, you may ignore this.

Project Identifier: CD7E23DEB13C1B8840F765F9016C084FD5D3F130

To find the starter code for this lab, go here.

3 Debugging

We have found in previous semesters that students entering 281 are not comfortable in their development environments, specifically with using git, gcc/clang, make, valgrind and debuggers. These tools and more are vital for success in projects and labs in 281, but also in future EECS courses you may take, and in industry. This assignment uses small toy programs to help you get more comfortable and hopefully learn a thing or two before jumping into more complex projects. **For each question, assume that the given code is compiled according to the flags given in the Makefile (shown below), under the CAEN Linux environment.**

Assume that all the programs are complete and that no other source files exist.

12. What is wrong with the following program?

```
int add(int a, int b);
int main() {
    int x = 1;
    int y = 5;
    return add(x, y);
} // main()
```

A. `add()` has no definition.

B. `add()` is called with `x` and `y`, but accepts `a` and `b`.

C. The program does not compile because `main()` cannot return a value of 6.

D. A function cannot be called after a `return` statement.

E. Nothing is wrong with this program.

13. What is wrong with the following program?

```
int main() {
    int x = 0;
    return x + y;
} // main()
```

A. `main()` cannot return the result of an arithmetic expression.

B. `main()` cannot return `x + y` because `x + y` is a `double`.

C. `main()` does not contain a declaration for `y`.

D. `main()` cannot return an integer.

E. Nothing is wrong with this program.

14. What is wrong with the following program?

```
#include <vector>
int *get_some_ints() {
   std::vector<int> ints = {1, 2, 3, 4, 5};
   return ints.data();
} // get_some_ints()
int main() {
   int *some_ints = get_some_ints();
   delete[] some_ints;
   return 0;
} // main()
```

A. The memory pointed to by `some_ints` is freed twice.

B. `main()` leaks the memory pointed to by `some_ints`.

C. A function cannot return a pointer.

D. `some_ints` is a pointer and not an array, so `delete` should be used instead of `delete[]`.

E. Nothing is wrong with this program.

15. What is wrong with the following program?

```
struct Thing { };
int main() {
   Thing a;
   Thing b;
   bool less = a < b;
}
```

A. `main()` cannot declare an instance of a `Thing` object.

B. `main()` must have a return value.

C. `Thing` cannot have an empty definition, so this code does not compile.

D. `main()` tries to use the < operator, which is not defined for the `Thing` type.

E. Nothing is wrong with this program.

16. What is wrong with the following program?

```
int main() {
   int some_ints[5] = { 1, 2, 3, 4, 0 };
   for (int *p = some_ints; p; ++p)
      *p = 0;
   return 0;
} // main()
```

A. `main()` indexes out of the bounds of `some_ints`.

B. `main()` leaks the memory pointed to by `some_ints`.

C. `some_ints` points to uninitialized memory.

D. The `for` loop is missing a curly brace, so the code will not compile.

E. Nothing is wrong with this program.

17. What is wrong with the following program?

```
#include <iostream>
int sum_ints() {
   int *some_ints = new int[50];
   for (int i = 0; i < 50; ++i) {
      some_ints[i] = i;
   } // for i
   int sum = 0;
   for (int i = 0; i < 50; ++i) {
      sum += some_ints[i];
   } // for i
   return sum;
} // sum_ints()
int main() {
   std::cout << sum_ints();
} // main()
```

A. `sum_ints()` indexes out of the bounds of `some_ints`.

B. `sum_ints()` leaks the memory pointed to by `some_ints`.

C. `some_ints` points to uninitialized memory.

D. `main()` must have a return value.

E. Nothing is wrong with this program.

18. What is wrong with the following program?

```
int factorial(int n) {
   return n * factorial(n - 1);
} // factorial()
int main() {
   return factorial(3) - 6;
} // main()
```

A. `factorial(3)` returns an uninitialized value.

B. A mathematical operation cannot follow a `return` statement.

C. `factorial(3)` never returns (and may cause a stack overflow).

D. `main()` cannot return the result of a recursive function.

E. Nothing is wrong with this program.

19. What is wrong with the following program?

```
#include <iostream>
int what_is_2x281() {
   int x, y = 281;
   return x += y;
} // what_is_2x281()
int main() {
   std::cout << "What is 2 x 281?\n" << what_is_2x281();
} // main()
```

A. The `+=` operator cannot be used after a `return` statement, since `x` would be returned before being added to `y`.

B. `what_is_2x281()` returns an uninitialized value.

C. The type of `y` is not defined.

D. `main()` must have a return value.

E. Nothing is wrong with this program.

20. What is wrong with the following program?

```
void takes_an_integer(int x) {}
int main() {
  size_t x;
  std::cin >> x;
  takes_an_integer(x);
} // main()
```

A. The code does not compile because `takes_an_integer()` has an empty definition.

B. `main()` must have a return value.

C. `takes_an_integer()` takes an `int` but is called with a `size_t`, which may cause a loss of precision.

D. `size_t` is not a valid type.

E. Nothing is wrong with this program.

4 File Input & Output

21. This code applies to the next two questions.

```
int main(int argc, char []*argv) {
   int i;
   // Read in an integer from a file.
} // main()
```

(a) Which line(s) of code would read an integer from a file using file redirection?

A. `ifstream readfile; readfile >> i;`

B. `cout << i;`

C. `cin >> i;`

D. `ofstream writefile; writefile << i;`

(b) Which of the following command line commands would run the above program (in `main.cpp`, already compiled to the executable `main`) with input file redirection?

A. `make all`

B. `./main input_file.txt`

C. `./main > input_file.txt`

D. `./main < input_file.txt`

22. A text file contains a list of words that are all on separate lines, with only a trailing newline after every word. One program, program `T1`, reads the file with `cin >>`, reading in to a `string`, `S1`. Another program, program `T2`, reads the file with `getline`, also reading into a `string`, `S2`.

(a) What will be the difference, if any, between the two resulting `string` variables?

A. `S2` will contain a trailing space, while `S1` will not.

B. Both strings will be exactly the same.

C. `S1` will have a newline character at the end, while `S2` will not.

D. `T1` will not read the file.

(b) What if there is a space at the end of each word in the file?

A. `S2` will contain a trailing space, while `S1` will not.

B. Both strings will be exactly the same.

C. `S1` will have a newline character at the end, while `S2` will not.

D. `T1` will not read the file.

5 Getopt

23. You're writing a simple text-based game on the command line. When the user runs your executable (`./281quest`) the game starts at the first level. If the user wants to skip to a certain level, they can instead run the program with the command `./281quest --level 5` (where `--level` is followed by the level they want to skip to). Which of the following correctly specifies this flag for `getopt`?

    A. `{"level", no_argument, nullptr, 'l' }`

    B. `{"level", optional_argument, nullptr, 'l' }`

    C. `{"level", required_argument, nullptr, 'l' }`

24. Short Options You're writing a program that will take five command line options: -p –paoletti, -d –darden, -a –angstadt, -m –markov, and -g –garcia. Options d, m, and g will have required arguments. The rest take no arguments. Which of the following strings is a correct short options string?

    A. `pd:am:g`

    B. `p:d:a:m:g`

    C. `p:dam:g:`

    D. `pd:am:g:`

    E. `p:da:mg`

