EECS402
Project Submission Instructions

All project submissions must be performed via the email submission process described in this document. Project submissions will NOT be accepted via Canvas or via emails to the lecturer, Instructional Aides (IAs), or graders. Since the submission process is an automated system, you must be sure to follow the specific directions exactly to avoid problems with your submission which could potentially result in your submission not being accepted.

It is ***your responsibility*** to submit your project correctly and verify that it was received fully as expected.

Building Your Project and Preparing For Submission

As described in lecture and/or discussion, we will use the "g++" compiler to build our projects in this course. When you compile your code for EECS402, you are required to use g++ and to make sure to include the required command line arguments "-Wall -std=c++98". So, for example, if you have a source file named "project1.cpp", you could build the executable using the command:

```
g++ -Wall -std=c++98 project1.cpp -o project1.exe
```

To repeat, it is required that you use g++ and that your use of g++ includes the -Wall and -std=c++98 command line arguments.

You'll also need to run the "valgrind" tool. When running valgrind, make sure you use the command line argument "--leak-check=full". This is likely not applicable for the first few projects, but it's a good habit to just always use that specific command line argument. For example, if you have an executable named "project1.exe", you can run your program in valgrind using the command:

```
valgrind --leak-check=full ./project1.exe
```

When you are ready to submit, you must also generate a file that shows you building your program (using the required command line arguments) and running your project in valgrind (using the required command line arguments). You will generate this file using Linux's "script" command, which, once started, will capture all keystrokes that you type in, as well as console output from programs you run. When on a Linux server, you start generating this file by simply issuing the command "script" at the Linux prompt. Next, you will build your project executable from source code, and demonstrate running your program in valgrind. When you are done, issue the command "exit" and the script command will stop recording your activity. All activity captured by the script command are stored in a file named "typescript". This file is required to be submitted along with the other files being submitted for the project. Please note: when you run your program with valgrind for the purposes of the typescript file, you do not need to demonstrate full testing of your project – it is expected that you have done that previously. Instead, just run your program in a somewhat quick and normal way such that it shows you used valgrind as required.

The entire process will look something like this. Red items are things that I typed in, black items are things that were output. The '$' before my typed commands is meant to indicate the Linux prompt, which may look different in your run.

```
$ script
Script started, file is typescript
$ g++ -Wall -std=c++98 project1.cpp -o project1.exe
$ valgrind --leak-check=full ./project1.exe
==30005== Memcheck, a memory error detector
==30005== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==30005== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==30005== Command: ./project1.exe
==30005==
<Your program's output will be here>
==30005==
==30005== HEAP SUMMARY:
==30005==     in use at exit: 0 bytes in 0 blocks
==30005==   total heap usage: 1 allocs, 1 frees, 57 bytes allocated
==30005==
==30005== All heap blocks were freed -- no leaks are possible
==30005==
==30005== For lists of detected and suppressed errors, rerun with: -s
==30005== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
$ exit
exit
Script done, file is typescript
```

Your output may look slightly different (i.e. the 30005 in the example output will be different when you run it, and the version of valgrind, etc., may be slightly different – if so, that's ok. Again, after you type the "exit" command, there will be a file in that directory literally named "typescript" – be sure to attach that file to your submission email as described below.

A quick note on the use of the "script": Once you start the script command, it will record every key you type – so, if you type some stuff and then backspace and then type some more stuff, those backspace characters will be encoded in the resulting typescript file, and make it "look" funny if you try to look at it in a text editor. You don't need to worry about this! We all make typos when typing and if your submitted typescript file has these weird characters in it, that is perfectly ok. Please make sure the typescript file you submit is a reasonably small size before you submit it. If your program can potentially produce a lot of output, make sure you run it in valgrind such that it doesn't produce too much output and the typescript file doesn't get submitted as a huge file.

Submitting A Project

You will submit your project by emailing your submission to a course account – NOT any of the course staff. The submission email address is:

eecs402@eecs.umich.edu

Please note the "eecs." part of the address, and make sure you compose your submission email using that exact address. Since the submission system is scripted, you must use a very specifically formatted subject line. All submission emails must have a subject line in the format

```
SUBMIT <#> <uniqname>
```

where the <#> is replaced with the current project number, and <uniqname> is replaced with your correctly spelled UM uniqname. For example, when submitting project 1, use the subject line:

```
SUBMIT 1 uniqname
```

NOTE: **Replace "uniqname" with your own uniqname!**

After ensuring the email address and subject lines are exactly as described, attach your project file(s) to the email. The files must be added as attachments – do NOT include any part of your submission or any information in the body of the message! The body of the message will never be read by a human. Also, do NOT zip, tar, bzip, or otherwise compress or combine files being submitted in any way.

Submitting Multiple Times

You may submit your project implementation as many times as needed. There is NO restriction on the number of submissions allowed. However, please understand these important caveats:

- We will ONLY consider your last received submission for grading. When a new submission is received, any previous submissions are disregarded – that is, even if one submission was successfully built, if your next submission has errors, we will only consider the most recent submission.
- Any files received in previous submissions will NOT be used during grading, so you must ensure that every submission you make is a *full and complete* submission. For example, if you have an implementation that requires 5 files and submit all 5, but then modify one of your files and want to re-submit, you must include all 5 of your files in the re-submission. Do not make any submissions containing "just the updated file(s)".

Resubmissions

During the resubmission phase, you are allowed to fix any correctness- and design-related deficiencies in your program and resubmit your project to potentially get some or all of those points back. The resubmission process is essentially the same as described above, except that you will append an "R" to the project number in the subject line. For example, if you received your grade for project 2 and had some correctness deductions taken, you can fix any bugs that you can and then resubmit the project using the subject line:

```
SUBMIT <#R> <uniqname>
```

where the <#> is replaced with the project number being resubmitted, and <uniqname> is replaced with your correctly spelled UM uniqname. For example, when resubmitting project 2 during the officially announced resubmission period, use the subject line:

```
SUBMIT 2R uniqname
```

NOTE: **Replace "uniqname" with your own uniqname!**

The rest of the submission process is the same as described in this document, so follow the same directions for resubmission as you do for the original project submissions, with the sole exception that you append an "R" to the project number in the subject line.

Feedback

The first three submissions on each day will provide feedback from the submission system about whether your project built successfully. We don't want students using this service as their compiler, so we are limiting to feedback on 3 submissions per day. After the first three on any given day, you may still submit your project, but you will NOT receive feedback about whether the build was successful (note: you should still receive an automated response (see below), but the response will not indicate the result of the attempted build). Obviously, you should be compiling/building your project yourself before submitting.

Restrictions

Please understand the following restrictions associated with the submission system.

File Types Accepted

Your submitted files may include:

- A file, named exactly “typescript”, generated by the Linux “script” command showing your project building correctly with the required command line arguments and showing your program running using valgrind using the required command line arguments
- C++ source code file(s) having extension ".cpp"
- C++ include file(s) having extension ".h"
- C++ “inline source” file(s) having extension ".inl"
- A file, named exactly “Makefile” for building the project via Linux’s “make”
- ***Note: Use of a Makefile, .h files, .inl files, and multiple source files will not be necessary until later projects, so they are not required or expected in project 1***

If unexpected file types are discovered by the submission system, they will be ignored, and may NOT be saved or considered in any way during grading – make sure your files have the proper names and/or extensions as described, or the files will be rejected.

Do NOT include any files that result from the build process. For example, do NOT include your project's built executable, or in later projects, the compiled object code (having extensions ".o").

Also, do not attach any compressed or combined files (i.e. do not zip, rar, tar, gzip, bzip, etc.) Submit your implementation files as separate, individual, attachments on the email.

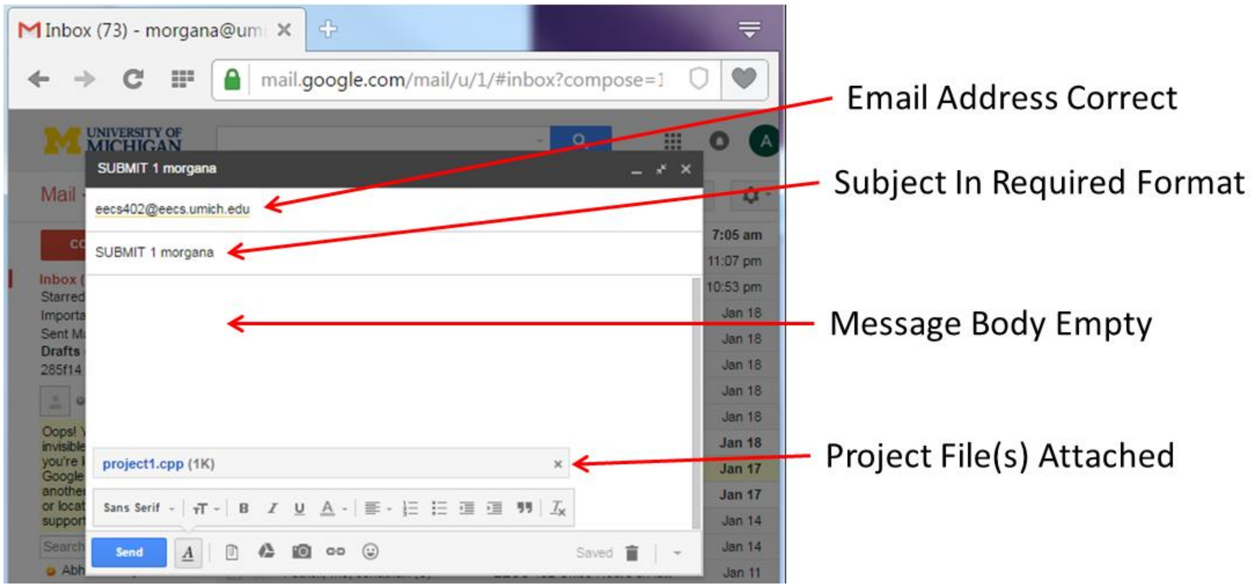
Multiple Mains

For projects that do not include a Makefile, you may only submit *one* main function implementation. Due to the way the automated system compiles your code, multiple main functions will not be accepted. For projects that do include a Makefile, multiple main functions are allowed as long as your Makefile handles them correctly. None of this paragraph applies to project 1.

Email Message Body

Do NOT include any information or source code in the body of the email message. The email is NOT read by a human and the body of the email message is not considered in any way during the submission processing. Any text in the body of the message will not be viewed, saved, or used in any way.

Example Submission



Automated Response

Under normal system load, assuming your submission was set up correctly, you should receive an automated response in around a minute or so.

If you do not receive a response, then your submission was NOT received by the system! It is YOUR responsibility to ensure you receive a response, and that you *read* the *entire* response to know the submission was completely successful.

Things you should closely check in your automated response:

- Early: Near the top of the response, it will indicate whether the received response is eligible for the early submission bonus or not.
- Number of Submissions Today: This will tell you how many submissions you have made on this particular day, and whether additional feedback will be provided if you're under the limit for feedback.
- Feedback: If you're under the number of submissions limit that provides feedback, additional feedback will be provided, including whether your submission was successfully built or not.
- Files Used / Not Used: A list of files that were found in your submission that were accepted files with proper extensions will be provided. You *must* make sure every single file that is important to your project's implementation is listed here. There is also a list of attachments that were found, but were NOT accepted as part of your submission. Look at this list as well to ensure you didn't accidentally submit a ".o" file in place of the ".cpp" file or something like that.

If you are very sure you used the correct email address and correct subject line, but get no response after 4 or 5 minutes, or if you get a response indicating something went wrong that was not related to your code, please post a message on Piazza so that the course staff can look into it.