Syllabus

Communication procedures (actually read and follow these)

- If you have a question regarding class administration or missing class, then please email the professor.
- If you have a question about a specific assignment, how an assignment was graded, or your grade tabulations, then please email the TAs/graders.
- If you have a question about theoretical course content, or assignment specifications, do NOT email, but instead, you should either:
  1. Come to the office hours of the professor or the TA. This option is guaranteed to at least get you an attempt at an answer.
  2. Post your question in the discussion forum in Canvas, which is monitored by the graders and the professor (with no guarantees about answers being provided, though we very much try). I encourage students to help others in the forum, with the exception that it would not be wise to post or copy code or assignment answers in the discussion forum.
- If you have a debugging or related question about your programming assignments, then attend either the TA's or professor's office hours. It is not feasible to fairly and equally satisfy all requests for programming or debugging help via email. Specifically, do not send your code to the TA or professor and expect that it is reasonable for them to debug it for you. Hint: this means you should plan to do your assignments early, to get debugging help if needed during office hours.

Professor

Dr. Taylor
http://cs.mst.edu/facultystaffandfacilities/facultydirectory/

Graders/TAs (Fall/Spring only)

Note: I will update who you should email for each assignment as we progress throughout the semester

- Mr. Essman <jberm6@mst.edu> (pa00, pa04, kq-trial week 1)
- Mr. Kovar <mjk8t2@mst.edu> (pa01, pa06, kq1-3)
- Mr. Scharf <msxfd@mst.edu> (pa03, pa07, kq-trial week 2)
- Mr. Bateman <ejbwn5@mst.edu> (pa02, pa05, kq4-6)

Course website

The course website is hosted here, at: https://web.mst.edu/~taylorpat/

Office hours and programming help

Here are some good sources of tutoring-style help in the class:

1. Instructor

Please feel free to come to my office hours, either during scheduled times or by appointment. If you are having trouble, this can be very helpful!

- Comp Sci room 212/213 (CS-Linux computer lab/lounge)
    - Times:
        - Starting the second week of class
        - Monday and Tuesday from 2:30-4
- Or by appointment (in CS 341)

2. LEAD hours (Fall/Spring only)

Also, there are LEAD learning center (http://lead.mst.edu/schedule/) hours.

- Comp Sci room 212/213 (CS-Linux computer lab/lounge)
    - Monday/Tuesday 2:00 - 4:00 pm CS 213 Taylor Johnathan Dunker LC
    - Friday 3:00 - 5:00 pm Comp Sci Lounge Taylor Reuben French Tr

The Learning Enhancement Across Disciplines Program (LEAD) sponsors free learning assistance in a wide range of courses for students who wish to increase their understanding, improve their skills, and validate their mastery of concepts and content in order to achieve their full potential. LEAD assistance starts no later than the third week of classes. Check out the online schedule at http://lead.mst.edu/assist, using zoom buttons to enlarge the view. Look to see what courses you are taking have collaborative LEAD learning centers (bottom half of schedule) and/or Individualized LEAD tutoring (top half of the schedule). For more information, contact the LEAD office at 341-7276 or email lead@mst.edu.

3. TA (Fall/Spring only)

Our graders (hours listed above), great students from previous semesters, will hold hours in the Lounge (Comp Sci room 212). They are available for help both with understanding the material, and also with the programming homework. If you can't make any of these times, but would like to schedule some programming help, please feel free to email any TA to set up a time.

Class/teaching evaluation and improvement

Please let me know what you like about the class and how it can be improved!

Course description

This course continues the development of structured programming concepts and their use in program development. Stacks, queues, linked list, arrays, trees, sorting, and searching will be taught together with their use in implementations of a number of algorithms. This course extends the study of structured programming by presenting the concepts and implementations of various abstract data types (ADTs) including lists, stacks, queues, trees, and graphs. We will discuss the trade-offs associated with implementing these ADTs with alternative data structures. You will also learn about more advanced C++ constructs.

Prerequisite

- Grade of "C" or better in Comp Sci 1570
- For a refresher on your C++

IntroProgramming:Content has detailed materials listed

- Remember the videos here: http://classes.mst.edu/compsci1570/
    - This C++ tutorial is pretty good, and well outlined: http://www.cplusplus.com/doc/tutorial

Course goals

"As gold which one cannot spend will make no person rich, so knowledge which one cannot apply will make no person wise." -- Samuel Johnson. As this quote suggests, the primary goal of this course is to provide you broad familiarity with methods commonly employed both in many real-world applications, and also those which are used for more advanced methods. The secondary goal of the class is to practice actually implementing these methods, both to assist understanding, and also to increase retention.

Textbooks

Required readings and activities will be assigned from these books:

Data Structures and Algorithm Analysis
Edition 3.2 (C++ Version)
Clifford A. Shaffer
Available FREE here: ../Content/DSA_Shaffer2013.pdf

Open Data Structures
C++ version
Pat Morin
Available FREE here: ../Content/ODS-cpp_Morin2018.pdf

Attendance

- Attendance will be taken indirectly (via daily web-surveys)
- Missing classes will greatly diminish your chances for getting a good grade in this class.
- If you miss more than 5 classes, we may drop you from the class.

Programming assignments

- You should expect around 1 technical

assignment every 1.5 weeks
- These will generally be due Tuesday night at 23:59, and assigned Wednesday night, with some exceptions

Working environment: the class virtual machine (current)

- Develop, test, and submit your assignments using the class virtual machine distributed as an OVA VirtualBox export file.
- Setup is detailed here: DataStructuresLab:Content:VirtualMachines
- We grade in this environment.

Campus computers (old, not guaranteed to work)

- Campus has Linux machines, and these will work for most purposes, but you must test in the provided VM before submission.
- Develop, test, and submit your assignments using the department Linux systems either in the lounge/lab, or remotely:
    - http://itrss.mst.edu/linux-support/
    - http://itrss.mst.edu/linux-support/user-documentation/

Remotely, ssh into the department's IT Linux systems (*nix):

> Where NN = computer number
>
> ```
> $ ssh yourlogin@rcNNxcs213.managed.mst.edu
> ```
>
> or for Windows, use https://putty.org to connect to the same address
>
> or if you want to run a graphical application (*nix):
>
> ```
> $ ssh -X yourlogin@rcNNxcs213.managed.mst.edu
> ```
>
> and for graphical applications on Windows:

- Run Xming
- Using https://putty.org enable X-fordarding under SSH menu: read ../../DataStructuresLab/Content/tools-for-computer-scientists.pdf Appendix B

Execution

Develop and test your program extensively

Compile with:

```
$ g++ file1.cpp fileN.cpp -std=c++11
```

Run with:

```
$ ./a.out
or
$ ./a.out <"sample_input.txt"
or
$ ./a.out <"sample_input.txt"
>"your_output.txt"
```

Check with:

```
$ valgrind ./a.out <"sample_input_if_exists.txt"
$ valgrind --leak-check=full ./a.out
<"sample_input_if_exists.txt"
```

Things you should check before you submit:

- Compare your output to any given sample output to make sure they are the same, including all newlines and spaces, via:

  ```
  $ diff --color sample_output.txt your_output.txt
  or for two-column format (easier to see):
  $ diff --color sample_output.txt your_output.txt
  ```

- Identity (name, SID) function included
- Files are in UTF-8, Unix delimited (which if you used solely the Linux environment without copy-pasting from Windows, you should be fine)
- Make sure to download and use the sample input and output text files we gave, not the text copied into a new text file you made.
- Your program compiles and runs in the specified Linux environment
- Check that file names match requirements and/or have not been changed
- Are you testing your execution with unchanged .h files if specified (by checking the file in an old commit), in case you edited for debugging purposes?
- Did you check the program with more input / output test cases than we gave you by

generating your own?
- Did you check that your functions all have the right inputs and outputs, even if they also have outputs to the screen?
- Did you push your latest commits and check you can see them online in the Gitlab interface?

Submitting your assignments via Git

To prepare for submitting assignments

1. Log into https://git-classes.mst.edu with your S&T login
2. Watch the videos here: https://git-scm.com/videos
3. Read Appendix E - Submitting homework with Git, in our Data Structures Lab manual: ../../DataStructuresLab/Content/tools-for-computer-scientists.pdf
4. Some optional extras include the full set of materials listed under the Version Control lab day here: DataStructuresLab:Content

Submit using the repositories created for each assignment at: https://git-classes.mst.edu/

Execute once:

```
$ git clone https://url-for-your-repository
```

Execute as many times as you like from within the directory/repository you cloned to your hard drive (just an example):

```
$ git status
$ git add *.cpp *.h *.hpp *.txt *.py
$ git add SUBDIRECTORY/*
$ git commit -m "Informative description of the commit"
$ git push
```

Do not add:

Compiled or generated files like a.out, your executable files, etc.
Put the name of these files in a text file named .gitignore

If you see your changes reflected on the git-classes site, you have submitted successfully.

If you work from different computers and want to synchronize, or we make changes to your repository:

```
$ git pull
```

Tips

- All of the functions in header files we provide must be completed as directed by the comments. Any changes to function names or parameters means we cannot properly grade that function, and you will not receive any points for it.
- Carefully read the comments of each member function, noting both return values and cout statements (they are not the same!).
- Develop your member functions one at a time, starting from the simplest ones.
- Move to the next function only after the previous one has been tested.
- Trying to code the whole class and then remove the bugs afterwards will likely prove to be too big a task.
- Build your own simple test cases.
- Print plenty of status messages to track the progress of your algorithm (and remember to comment them out before submission)
- Using MS Visual Studio or online web-based compilers is bad idea... Stick to GCC and/or local IDEs that are g++ compatible

Quizzes / Daily questions

- Published experimental studies in the fields of research in cognitive psychology and education have shown that frequent (rather than sparse) recall, is both more effective for learning, retention, and synthesis, and also encourages frequent smaller bouts of studying, rather than cramming.
- To incentivise regular reading and attendance, we will have daily quizzes. These are administered using a clicker-like system. To avoid having to pay for a clicker, we use a free service called Kahoot (https://en.wikipedia.org/wiki/Kahoot!). You will need to have a web-capable device in class (android, iphone, laptop, chromebook, etc.). This is a fair expectation for the following reasons:

  - Most students have a smart-phone or laptop.
  - If you do not, classes already require students to pay for a clicker ($40+), and one can obtain a web-capable Android smartphone (a.k.a. prepaid burner) for less that $20 at local stores such as Kroger, Walmart, and most gas stations, without a service plan (merely need wifi, which is free on campus).

Grading

You will be graded based on assignments, projects, homework, and other miscellaneous activities.
We reserve the right to factor in points for attendance related performance.

Assignment grading

- Programs will be graded (on a scale from 0 to 100) primarily on their correctness.
- Complete and correct output for every test input case is necessary for a full score.
- An entirely non-compiling, non-running, or crashing (Seg-fault, core dump, etc) program or script will receive a score of 0. We design unit tests so that one can crash and the rest can succeed, so that you can get more points.
- If a program compiles and runs, then points will be deducted for each incorrect test case output. Points may also be deducted for:
  - Missing name function
  - Incorrectly formatted output. (Presentation Error)
  - Memory Leaks
  - Specific types of inefficiency
  - File format issues

It is expected that all of your work runs correctly in the specified Linux environment we are working with in class, in the exact manner we specify in the assignment description. If you were contracted to write code for a job, and it ran on your computer, but not your employer's as they needed, your work would be considered a failure. In that light, you are also responsible for submitting all text and source files encoded UTF-8, Unix delimited.

The test cases we will run for grading are more extensive than any sample input we give you. It is possible, even likely, that if your program seemingly works perfectly, for example with a sample_input.txt, that it may not work perfectly with our grading; this is fair and reasonable challenge, since we describe the bounds of performance required generally; when coding in the real world for a job, you will be expected to anticipate edge cases, weird behavior, larger than expected datasets, etc. Practicing this can help you train one of the more important skills of an industry programmer. You should make some test cases yourself, that have input and output, perhaps different or which exemplify some edge case.

We do not currently grade on your style, but highly recommend reading the MST-CS style guide on the syllabus. Good programmers don’t always have good comments, but they almost always have clean, consistent, readable code style and formatting... Please read, and re-read the department's C++ coding standards: http://web.mst.edu/~cpp/cpp_coding_standard_v1_1.pdf

After grading any given assignment, if the assignment appeared to be too difficult for the class, we may normalize to the top student's performance (the student with the highest point rank will get a 100% / A). This can, by definition, only help your grade, but not hurt it.

Quiz grading

- Our free web-based "clicker"-like questions will be treated as daily quizzes.
- They will not be graded for the first week of the semester only (all following weeks will be graded).
- Time does not count for points (though it does in Kahoot scoring), just correctness.
- If you miss class, you will miss the points for that day.
- There are several ways to do well on these quizzes:
    1. Come to class.
    2. Do the reading on the topic to be lecture, BEFORE CLASS.
    3. Come prepared with your web-capable

device.

4. Make sure to use your correct assigned "anonymous" user-code (if you do not, you will not get points for the day).

Overall grading

We grade using the following procedure (percentages for each category may change slightly):

- Assignments (94% of your grade):
  - 100 points for each technical assignment
  - 150 points for the final project
- Daily Kahoot quizzes (6% of your grade)
  - 3 points for each quiz (usually 3 questions per day).
- Any miscellaneous points

Your final grade = percent of possible points above
Your letter grade = standard S&T letter-percentile mapping:

A : [90.00 - 100] %
B : [80.00 - 90) %
C : [70.00 - 80) %
D : [60.00 - 70) %
F : < 60 %

Grades will not be rounded; for example, if you have a 79.9, that is a C.

View your grades

You can check your grades on Canvas:
http://canvas.mst.edu/

Makeup and late work

- No technical assignment re-submission is allowed, except on the first assignment under limited circumstances.
- Late submissions will not be accepted. It would be wise to account for something unexpected popping up last minute, so try to finish early.
- If you have an S&T-acceptable documented reason (i.e., illness, death in the family, etc) for missing **in-class** events, please see the professor to discuss potential re-scheduling or accommodation.

Academic honesty

You're here to learn and better yourself! Write all your work in your own words, and write your own code. Do not copy-paste (plagiarize) from any source. If you are not sure, err on the side of caution and do your work independently. Occasional infrequent help from a friend when your are really stuck may be reasonable, though if that "help" is frequent enough that your collaboration results in almost identical code, it was too much collaboration for an assignment intended to be independent work (which all are unless explicitly assigned as group work).

If you are found to be engaging in any form of academic dishonesty, the most severe penalties permitted by the university will be enacted. Incidences will typically result in grades of 0 for the respective course components, as well as notification of the student's advisor, the student's department chair, and the campus undergraduate studies office. Further academic sanctions may be imposed as well in accordance with university regulations (http://academicsupport.mst.edu/academicintegrity/). Those who allow others to copy their work are also committing plagiarism and will be subjected to the same procedures.

The Honor Code can be found at this link: http://stuco.mst.edu/honor-code/. Page 30 of the Student Academic Regulations handbook describes the student standard of conduct relative to the University of Missouri System's Collected Rules and Regulations section 200.010, and offers descriptions of academic dishonesty including cheating, plagiarism or sabotage (http://registrar.mst.edu/academicregs/index.html). Also see: http://academicsupport.mst.edu/academicintegrity/studentresources-ai

We check your assignments against each other with software that is VERY good at detecting similarities and differences between any text files, including your source files. These methods are difficult, if not impossible to trick. Please do not try to copy-paste, share sources directly, or write all your code in a group or pair for individual assignments; you will not like the consequences!

Attempting to deceive attendance checking procedures is considered academic dishonesty for ALL parties involved. For example, do not submit someone else's pre-lab or lab assignment for them because they are not attending class.

Burns & McDonnell Student Success Center

The Student Success Center is a centralized location designed for students to visit and feel comfortable about utilizing the campus resources available. The Student Success Center was developed as a campus wide initiative to foster a sense of responsibility and self-directedness to all S&T students by providing peer mentors, caring staff, and approachable faculty and administrators who are student centered and supportive of student success. Visit the SSC at 198 Toomey Hall; 573-341-7596; success@mst.edu; web: http://studentsuccess.mst.edu/

Accessibility and Accommodations

If you have a documented disability and would like accommodations in this course, please facillitate providing documentation to the professor as early as possible in the semester. Disability Support Services staff will need to send a letter to the professor specifying the accommodation you will need. It is the university’s goal that learning experiences be as accessible as possible. If you anticipate or experience physical or academic barriers based on disability, please contact Student Disability Services at (573) 341-6655, sdsmst@mst.edu, visit http://dss.mst.edu/ for information, or go to mineraccess.mst.edu to initiate the accommodation process. Please be aware that any accessible tables and chairs in this room should remain available for students who find that standard classroom seating is not usable.

Title IX

Missouri University of Science and Technology is committed to the safety and well-being of all members of its community. US Federal Law Title IX states that no member of the university community shall, on the basis of sex, be excluded from participation in, or be denied benefits of, or be subjected to discrimination under any education program or activity. Furthermore, in accordance with

Title IX guidelines from the US Office of Civil Rights, Missouri S&T requires that all faculty and staff members report, to the Missouri S&T Title IX Coordinator, any notice of sexual harassment, abuse, and/or violence (including personal relational abuse, relational/domestic violence, and stalking) disclosed through communication including but not limited to direct conversation, email, social media, classroom papers and homework exercises. Missouri S&T's Title IX Coordinator is interim chief diversity officer Neil Outar. Contact him (naoutar@mst.edu; (573) 341-6038; Temporary Facility A-1200 N. Pine Street) to report Title IX violations. To learn more about Title IX resources and reporting options (confidential and non-confidential) available to Missouri S&T students, staff, and faculty, please visit http://titleix.mst.edu.

Classroom Egress Maps

http://designconstruction.mst.edu/floorplan/

Backlinks: Index:DataStructures CoursesArchive:DataStructures SS18:Content Index:DataStructures:Content