McMaster Engineering Competition 2018

Programming

October 20th 2018 - October 21st 2018

Challenge Statement

Over the past century, programming has become a very powerful tool utilized in endless applications. Alan Turing's Enigma machine was pivotal in changing the tides of war. The introduction of the world wide web, and with it Amazon, Google, and various infamous sites have changed the way the world accesses information. It is evident programming, in more ways than one, has changed the way the world operates, but it has also changed the lives of the less fortunate for the better. Domain-specific programming and embedded computation have the ability to give disease diagnosed individuals a chance at independence again.

Victims of paralyses diseases almost always result in the individual losing their ability to speak along with their diminished physical abilities, think Stephen Hawking due to his Amyotrophic Lateral Sclerosis (ALS). This is why interactive communications software must be made to help said individuals relay and convey messages locally to others. Engineers have the ability to revive lives through these simple embedded applications.

Deliverables

Your solution must contain the following deliverables:

1. **Program:** Your team must design and develop a program that contains the technical requirements stated in the following subcategory. The program at the very least will consist of a graphical user interface and with it an input and output system. The system must be self-standing come demoing time and should be a unit-under-test, thus the test bench cannot be incorporated into the program itself.

2. **Demo:** The program must be demoed at the end of the competition period. The demo should display the input and output methodology. The judges are free to ask specific commands during this session so be certain every possible test case is covered during your team's testing. The demo period is also an opportunity to communicate aspects of your program you believe the judges should know. For example, given more time what else could you have been able to accomplish, or what essential loose ends still need completing. Later on, we will discuss why the potential of your program can be of use.

3. **Documentation:** Your solution must be submitted with its accompanying software documentation. The documentation behaves like a guide to your program, allowing any programmer to understand your code without author interference. It is not necessary to follow industry protocol; As long as the eccentric aspects of the program are covered. For example function definitions, I/Os, **bugs**, error handling etc. Also include the reference to all of the code that is not yours, such as libraries, APIs etc.

A concrete laid out presentation accompanying the demo is not necessary however it is recommended. Should your team qualify for the next round, you will be presenting your solution in front of industry judges. Having a well laid out presentation outlining the highlights of your design could be the difference between first and second place! The presentation would not have to be too detailed, simply outline your team's ideologies and thought processes, using your documentation as a visual aid.

You must submit all deliverables at the end of the competition period, note that only the submitted resources will be allowed in your presentation, given your team qualifies. During the competition time, receiving any aid from a non-team member via messenger or in person will result in **instant disqualification** (further covered in rules)**.** Same goes for the use of unsubmitted sources during the Sunday presentation.

Your documentation must be in PDF, MS Word, or Google Document format, whilst any presentation must be in made in Google Slides or MS Powerpoint should your team wish to do so. All documents must be formal and use appropriate formatting.

Objectives, Requirements and Constraints

Don Loree

Don Loree, brother of Bob Loree (retired Director of Engineering 1, Mac Alumni, THE BLUE LOUNGE) is going to be the recipient of a communication device which your team will be designing/developing. Don is diagnosed with **Progressive Supranuclear Palsy (PSP)**, which inhibits his physical movements and reduces his vision. Don spends his time in a hospital bed or wheelchair in a laid back position. Amongst other things, Don has also lost his ability to speak, he can however hear and also minimally use his hands. Thus, if a device with a touchpad, say for example a Microsoft surface, were to be mounted on his wheelchair or next to his bed, a onboard program catered towards Don could behave as his communication device. Prior to competition end, Bob Loree is willing to work with the brightest team(s) and see through it that their program is fully completed and polished for Don to use! This is a fantastic opportunity to not only make a real change but to do it using the programming knowledge you've been sharpening and honing to this day.

So above is an image of Don, as you can see he has functionality with his hands. A lot of questions have been asked about Don's physical prowess; Don cannot move his head, however, the hardware given to him would be placed in his field of vision, thus he would be able to use it. Just focus on the program and the objective and do not worry too much about the hardware or don's physical abilities.

The primary objective of the challenge is to design a communication program for Don. To make the challenge less broad we want you to follow the following technical requirements, aimed towards making a program that would be most effective for Don:

- **INPUT:** The input must be a large keyboard, however, this will not be any ordinary keyboard. To make Don's typing speed faster, you must implement a predictive text keyboard. Predictive text can be found on latest generation devices. It is the process by which the keyboard predicts future words based on the characters typed thus far and displays them to be appended onto the sentence. You may design your own predictive algorithm or use an online library or API. The physical layout of the keyboard as well as how the recommended words are displayed is up to your team.
- **OUTPUT:** The output must ideally be a text to speech voice. It will use Don's input sentence, thus vocally communicating Don's thoughts. Similar to the input, you will be allowed to use online APIs to implement the text to speech module. Note that the text to speech voice must only activate once Don indicates it, not every time a word is written. This indication flag can be whatever your team prefers.
- If time permits, you are encouraged to extend your output mechanism to a non-local channel such as a wifi based chat room or messenger system. This would lead to a great increase in scoring.

There are various ways to develop the objective stated above. We want to make sure that the code you write is well structured the following sub-requirements will also be taken into account when judging:

- Efficient/Space: Your space and time complex cannot be infinity or exponential. We don't want a program that lags or has noticeable latency.

- Portable: the code must be wrapped up in an executable or shareable format of your choosing. Delivering your program as well as ease of use are an important aspect of programming, the user should not need to compile code or open multiple files.
- Modular: This ties in with efficient/space as making the code modular will automatically make it more efficient. Use of object-oriented programming, functions, etc. will lead to less repetitive code, easier debugging in the future, and flexible code.

Note: your system does not have to be complex. Functionality is key meaning make meeting the requirements your top priority. Once you have met the requirements, your team is free to add extra features. Some ideas include quick access buttons to more common phrases, for example, one of Don's more common phrases is "Oh yeah". You could also include an "emergency" protocol button.

We understand that groups could have great ideas beyond the scope of the requirements, thus if you or your team wishes to embark on a different input or output method, ask the volunteers or coordinator beforehand. Once you have been given approval you may implement your desired design. Your requirement adjustments will be taken into account for the rubric.

For some groups, the requirements may be too demanding in the limited time frame, but do not let this diminish your efforts! The key to a good solution is proof of concept, if your team is able to build a good foundation upon which a sophisticated piece of software is perceivable, then you are still a qualifying contender even with the diminished possibility. At the end of the day we want you to create a solution that Don can physically use. This does not mean you should negate making an actual prototype as a working solution is still favored.

Tips

Get to know each other! Communication and acknowledgment of each others skill sets will allow your team to allocate work in an efficient manner. Working to your groups' strengths will automatically ensure your team will provide a good piece of software in the given time!

Don't feel outclassed! Your program doesn't have to be the most intricate, or the most complex. You as well as the judges know there are only 6 hours to compile the entire project. There will be flaws and that is ok! The purpose of the challenge is to work at the best of your abilities and test yourselves in a possible real-life scenario, so give it your best shot.

Document as you go! Documentation can seem very tedious and leaving it as the last step may appear like the correct choice, however, this is arguably wrong. Documenting and commenting can enable your group to catch possible errors and keep track of the progress. In deeper stages of coding, you have easier access to your code due to the already organized fashion it will be in. Finally, you will effectively conserve a lot of your final hour!

Rules

1. You are allowed access to internet during the competition period as well as online libraries, packages, APIs etc. Upon using any code that was not yours, you must reference it and mention it during your demoing. Plagiarizing entire codes will result in disqualification so avoid it at all costs.

2. All deliverables must be made during the 6 hour competition period. Also, only the material submitted before competition period end can be used in your presentation meaning you may not bring any external resources going into presentation day.

3. You are not allowed to accept aid from anyone outside of your team. This includes having someone come in to help or substitute with an existing member.

4. You are not allowed to use any personal devices or other online communication methods (instant messengers, iMessage, email, etc.). Social media sites and any other communication tools are prohibited.

5. All questions regarding the case must be asked before the competition start, so during welcome and briefing session. During the competition period the coordinator, as well as volunteers, will be limited in terms of what questions they can answer.

6. Not submitting your solution and its associated files will result in a penalty so be sure begin the assembling process ~15 minutes prior to competition end.

Violation of any of the above rules can lead to penalties, disqualifications or other means of deduction.

Permitted Tools

Physical Tools

During the competition period you will be provided with a designated workspace in which you may use:

- Computers (max one per person)
- Peripherals (monitors, mice, headphones, keyboards etc.)
- Monitors (max 2)
- Reference material (textbooks, course notes)

Software Tools:

It is expected that teams participating in this competition have adequate knowledge in choosing what tools to have prepared for competition day. That being said there are no restrictions in terms of what software tools are permitted however we do recommend the following languages, IDEs, text editors, and databases:

Languages:

- HTML/CSS/Javascript
- Jquery
- Java
- C/C++
- Objective C
- PHP
- Ruby
- .NET (C#, VB, VBscript)
- Python
- Perl
- Actionscript
- Coffeescript
- Scala
- Swift
- SQL
- MongoDB

IDE

- Eclipse
- NetBeans
- Microsoft Visual Studio
- JetBrains (IntelliJ, PyCharm, WebStorm etc.)
- XCode
- Android Studio

Text Editors

- Sublime
- Atom
- Vim
- Notepad++
- Kwrite
- Visual Studio Code
- Bracket

Other Tools

- ReactJS
- MeteorJS
- NodeJS
- Bootstrap
- Flask
- MEAN Stack
- Digital Ocean
- Heroku
- Microsoft Azure
- Amazon Web Services
- Ionic
- Codava
- Ruby on Rails

Resource:
If your team is having trouble finding common ground with the challenge or are absolutely lost, then consider using the following package!

https://pypi.org/project/autocomplete/

This is an MIT Licensed software developed by Rodrigo Palacios. The page covers how the algorithm is implemented as well as an explain like I'm 5 section!

If you choose to use the autocomplete package, you will be better off also creating the GUI on a python framework. Look into the following resources as they are well-known meaning you will be able to find a lot of reference material online:

Tkinter **Kivy**

PyQt **WxPython**

RUBRIC

Criteria	Unsatisfactory	Satisfactory	Good	Excellent	Total
Functionality	0-15	16-34	35-49	50-60	
	- Input keyboard does not work - No predictive text available, no words appear upon typing - No output form present at all - lag and latency - covers no non expected inputs	- Input keyboard works - No predictive text available, no words appear upon typing - Text to speech not implemented, no other output form present - noticeable lag and latency - covers some non expected inputs	- Input keyboard functions - Predictive text words appear most of the time - Text to speech not implemented, but text output is present - Little lag and latency - covers most non expected inputs	- Input keyboard functions seamlessly - Predictive text words well laid out and work majority if not all the time - Text to speech, other form of output works upon trigger - Little to no lag or latency - Covers all non expected inputs	/50
Feasibility	0	1-14	15-24	25-30	
	- Don would have a hard time using the program - GUI is too small/very hard to maneuver - Keyboard placement is poor	- Don would have some issues using the program - GUI is small/some difficulty to maneuver - Keyboard placement is adequate	- Don can use the program majority of times with no issue - GUI is well laid out - Keyboard placement is good	- Don would have no trouble using the program - GUI is very well laid out - Keyboard placement is great	/30
Code Structure/ Documentation	0	1-9	10-14	15-20	
	- Little to none commenting in code - Documentation is not submitted	- Code is commented vaguely - Documentation is not very detailed - missing crucial code	- Code is commented - Documentation is easy to read - Contains most of the code contents	- Code is well commented - Documentation is detailed and easy to read - Documentation contains all of the major codes	/20
Bugs/Penalties	Deduct	10 - 15	0 - 9	0	-
		Many bugs found, program not usable	Few bugs found when running progra6m	No bugs found or non detrimental bugs found	
Bonus	Add	4 - 7	8 - 11	12 - 15	-
		- 1 extra useful feature added	- 2 extra useful features added	- 3 extra useful features added OR implemented a messenger system for the output	
					Final: ______ /100

Weekend Schedule

Day 1 - Saturday October 20th, 2018

*BSB = Burke Science Building

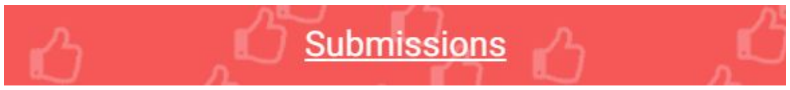
10:45 am	Sign-in period	BSB Main Lobby
12:00 pm	Opening Ceremonies	BSB B136
12:45 pm	Introduction to Challenges	BSB 244, 249, 115
1:00 pm	Challenges begin	BSB 244, 249, 115
5:00 pm	Dinner break	BSB 244, 249, 115
7:00 pm	Challenges end	BSB 244, 249, 115
7:15 pm	Preliminary round of judging to determine finalists	BSB 244, 249, 115

Day 2 - Sunday October 21th, 2018

9:00 am	Check-in w/ Breakfast, coffee etc.	BSB Main Lobby
10:00 am	Finalists presentations to Industry Judge panel	BSB
12:00 pm	Networking period	Celebration Hall
1:00 pm	Lunch break	Celebration Hall
1:30 pm	Closing Ceremonies	Celebration Hall

SUBMISSION INSTRUCTIONS

1. Visit https://mec.macengsociety.ca/
2. Click on the 'Submissions' bar located at the top



3. Fill out the form (only one member has to)
4. Make sure to attach your program files, documentation, and any other tangible files
5. Submit!

If your files are too big, use a github repository and link it. Submit a text file with the link to your repository. Name it "GitHub.txt".

Sahil Patel and Ian Currie will be coming around after 7:00 to view your demos. Do not work on your solution after 7:00, there will be invigilators. Finally note that late submissions are prone to penalties.