WATERLOO ENGINEERING

WEC

Programming Competition

Competitor Information Package

Fall 2021

Waterloo Engineering Competition

November 13, 2021

SCHEDULE

The schedule of the Spring 2021 *Programming* competition is as follows:

Time	Activity	Platform
08:45 – 08:55	Sign-In	Zoom
09:00 – 09:15	Opening Ceremony	Zoom
09:15 – 09:45	Competition Introduction & Questions Period	Zoom (Breakout Rooms)
09:45 – 15:45	Design	Self-Administered
15:45 – 16:30	BREAK	N/A
16:30 – 19:30	Presentations & Judging	Zoom (Breakout Rooms)
19:30 – 20:00	Judges Feedback & Awards	Zoom

*Times are approximate. Exact event timing may change on the day of the competition, which will be communicated in the WEC S2021 Discord in #*announcements*.

GENERAL RULES & GUIDELINES

1. The Welcome and Competition Introductions shall be hosted via a Zoom call. The joining information will be emailed to the competitors and posted in the #announcements and *#programming* channels in the Discord.

2. Competitors will be presented with a 15-minute question period following the welcome and briefing. Please try to ask your questions during this time period so that you have full WEC staff attention. Beyond this, if questions arise, you may ask them in the discord server in your respective channel. You can access the discord channel through this link: https://bit.ly/DiscordWEC. Competition questions will only be answered in the *#programming* channel.

3. The design and build stage is **six (6) hours** in duration. Teams may use any video or voice conferencing platform they are comfortable with. Please ensure that the platform supports screen sharing to promote collaboration. The organizers will be active on Discord for any support or questions needed. If a voice/video call with the Director is required, the Director may join the team's ongoing call, or they can set up a new call on a platform of their choosing.

4. All deliverables must be submitted to the form prior to the end of the design and build stage. It is the team's responsibility to submit these deliverables using the proper naming convention by the end of the allotted time. **Teams will receive a penalty (6%) for late submissions. If a team is three (3) or more minutes late, the team will be disqualified from competition.**

5. If teams are unsure about rules or require further clarification, please ask one of the organizers. Volunteers may be able to assist, but in the event of discrepancies between volunteers and organizers, the organizers' opinion will be followed.

6. Time remaining in the competition will be announced to competitors at the 3-hour, 1-hour, 30-minute and 10-minute marks.

7. If the discovery of a rule violation occurs during the competition, immediately following the competition, competitors have 24 hours to initiate an appeal for the decision. If the discovery of a rule violation occurs after the competition, the competitors will be immediately notified via email and will have three (3) days to appeal the decision. Should a competitor fall into one of these situations, the competitor should contact the WEC Commissioner.

JUDGING INFORMATION

1. Judging Schedule will be released at 16:00, with the first presentation slot occurring at 16:30.

2. During the Presentations & Judging stage, the judges will be on a Zoom call in their respective breakout rooms for the duration of the stage. Teams should join the Zoom call 5 minutes before their time slot. Teams will be given direction on when to enter the breakout room for judging. Teams will present their own slides during their slot, but in case there are technical difficulties the judges will also have copies of the slides.

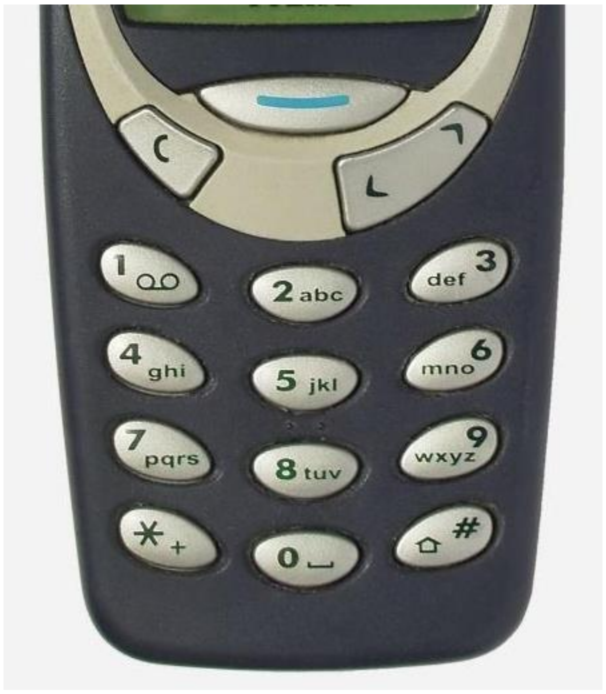
3. After the judge question period, the presenting team shall leave, and judges will be given a 5-minute consolidation period before the next team presents.

4. After all teams have presented, the judges will be given 10 minutes to deliberate among themselves to compare evaluations and determine winners. Teams should be ready to join back into the meeting for the judges' feedback and announcement of winners after all teams finish presenting.

SCENARIO

The University of Waterloo has an intelligence agency and student agents need to use untraceable phones to communicate using code words. Keypad phones are the only ones that can be used to communicate. For an agent, time is a very precious commodity in high pressure situations, so agents need to be able to communicate their status fast using code words. As part of the IT department, you have been given the task of creating a program which can output words that can be typed in the shortest amount of time. This program will help agents to choose code words wisely for faster communication…..

DETAILS



How a keypad phone works:

In a traditional keypad phone, typing letters is more time-consuming than with smartphones. In the picture above you can see how a traditional keypad looks like. This is a short explanation on how words are typed into it:

1. There are buttons labelled 0-9 and each number has three or four letters associated with it. Number 0 has ‘SPACE’ associated with it.
2. One tap on any of the buttons will print the 1st letter associated with that button. For example, pressing ‘2’ once will print ‘a’. To get the consecutive letters like ‘b’ and ‘c’, ‘2’ needs to be pressed twice and thrice respectively.

3. To print ‘b’ after printing ‘a’ or any such case where you need to print two letters consecutively associated with the same number, you need to wait till the system registers no further taps are coming in. For example, to print ‘ab’, you press ‘2’ once (outputs ‘a’), wait, then press ‘2’ twice (outputs ‘b’).
4. For capital letters, you need to get to the corresponding small letter by the above methods, then hold the button for that letter. For example, for capital ‘C’, tap button ‘2’ thrice and on the third tap, hold it till the letter becomes capital.

Time intervals between key taps:

1. The timer starts only after the **first** button is pressed.
2. Whenever you move to a **new** button, the time interval between the 2 button presses is assumed to be **0.25s**. Ex: If you press ‘3’ once to get ‘d’, and directly after that you press ‘4’ once to get ‘g’, the time interval between the 2 is 0.25s.
3. To get any letter apart from the first under the same number, say ‘b’ or ‘c’ under the number 2, you need to tap the key after an interval of **0.25s**. So, if you want to get ‘b’, you tap 2 twice and between those taps there is an interval of 0.25s. To get ‘c’, you tap 2 thrice with a 0.25s interval between those taps. Thus, to print just ‘c’, it takes 0.5s.
4. To type the consecutive letters under the same number, say ‘ab’ under the number 2, you need to wait for **0.5s** before the system registers the first letter ‘a’. After that you press 2 twice following rule 3 above to get ‘b’. Thus, to get ‘ab’, it takes 0.5+0.25 = 0.75s.
5. To get capital letters, you need to get to the corresponding small letter (by above methods) and then hold the button for **2s** (instead of 0s for a key press). Ex: If you want to get capital B, you will press 2 twice tap (time interval = 0.25s, refer point 3) and on the second tap you will hold the key for 2s. Thus, in total it will take 2.25s to get the letter B.
6. For keys apart from 2-9, if a functionality requires consecutive taps for that key, the time interval between those taps will be **0.25s** like Rule 2 above.
7. If you need to hold a key to implement any functionality, it will have to be for **2s** similar to Rule 5.

Example words and their time calculation:

- **dEFine:**
  d: 0s (Rule 1, first letter takes 0s)
  dE: 0 + 0.5 (Rule 4) + 0.25 (Rule 3) + 2 (Rule 5) = 2.75s
  dEF: 2.75 + 0.5 (Rule 4) + 2 X 0.25 (Rule 3 applied twice since ‘f’ is the 3rd letter under 3) + 2 (Rule 5) = 5.75s
  dEFi = 5.75 + 0.25 (Rule 2) + 2 X 0.25 (Rule 3 applied twice) = 6.5s
  dEFin = 6.5 + 0.25 (Rule 2) + 0.25 (Rule 3) = 7s
  dEFine = 7 + 0.25 (Rule 2) + 0.25 (Rule 3) = 7.5s

- **Crisp:**
  C: 0.25 + 0.25 (Rule 3) + 2 (Rule 5) = 2.5s
  Cr: 2.5 + 0.25 (Rule 2) + 2 X 0.25 (Rule 3) = 3.25s
  Cri: 3.25 + 0.25 (Rule 2) + 2 X 0.25 (Rule 3) = 4s
  Cris: 4 + 0.25 (Rule 2) + 3 X 0.25 (Rule 3) = 5s
  Crisp: 5 + 0.5 (Rule 4) = 5.5s

OBJECTIVE, REQUIREMENTS & CONSTRAINTS

Part 1:

Given the default rules above, in your test files you will be given a maximum of 10 different words of any combination of letters (small, capital). Note each test file will have words of the SAME LENGTH you must determine which word (or words) can be entered into the phone in the least amount of time. **NOTE**: I say word**s** because 2 or more words can take the same time to type!
This is a matter of you implementing the rules as described above correctly. You will print all the words from the file that have the minimum time in the order in which the words appear in the file. Once you have printed all the words, you will print the time to enter the word in seconds, so for the two above examples, you would have outputted 7.5 and 5.5, respectively.
For this component, we will require both the program together with a design document that describes how your program finds that fastest algorithm. You may include state diagrams, flow charts, etc., to explain how you find the time. Your design document should try to justify why your design does return the correct time.
IMPORTANT NOTE: These test files will have 'windows end of line' characters. So please create your parsing code in accordance with that. Some programming languages handle that automatically whereas others do not.

Part 2:

You need to add the functionality of what happens if a button is broken. The new set of text files will contain which button is broken in the first line.
NOTE: Only one button will be broken at a time.
Now if a button is broken it doesn't mean you ignore those words that contain those letters. This is the section where you get to use your engineering skills and creativity to think of how you can use the other buttons to compensate for the broken one. If you chose to use the other

available buttons, you need to think of a way to retain the functionality that the button provides.
Below is a list of the functionalities of the buttons:

- Buttons 2-9: Typing letters
- Button 1: Voice Mail
- Button *: To print *
- Button #: To print #
- The button with the call icon: Accept calls when pressed once
- Arrow buttons: To go back and forth between pages

Based on this, your results will change. The words in the text file will still be of the same length. Look at DELIVERABLES below for a more detailed understanding of how and what you need to submit.

Part 3:

How would you better reprogram the buttons to type words faster given that you can only have the same number of buttons? This is not a programming question. Imagine a period when there was no BlackBerry or even an iPhone. How would you arrange the letters or use the 12 buttons in a different way to make typing faster? This question purely tests your creativity and marks will be given based on that. See DELIVERABLES below to understand how to submit this question.

DELIVERABLES

For Part 1:

You will need to parse each word of an input text file, calculate the time for all of them and output the ones which would take the shortest amount to type with the time it will take to type them.
The design document is also a deliverable in this part. As mentioned above, it can be flowcharts, state diagrams etc. enough to explain why your algorithm calculates the word(s) which will take the shortest time to type.

Sample Input .txt file:

```
adgjm
dePOR
PotAT
NASAs
cRIed
wtpmj
```

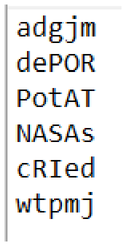
Sample Output(s):

```
adgjm = 1s
wtpmj = 1s
```

For Part 2:

1. You need to provide a solution on how you would compensate for the broken button. That solution can be in any form (written, flowchart) enough to convey the information. If you choose to use any of the other buttons, an explanation needs to be added on how you would retain the existing functionality of that button. This can be done in many ways and it is up to you to decide.
2. Once you have laid down your solution, you will need to implement that in your program. The time interval rules stay the same as listed above in DETAILS. Your program for part 2 with the added condition for a broken button will do the same thing as PART 1. It will Output the word(s) which will take the shortest amount of time to type.

Sample Input .txt file:



```
adgjm
dePOR
PotAT
NASAs
cRIed
wtpmj
```

Sample Output(s):

In this situation, the sample output will be different for everyone based on the solution they come up with.

3. Since the output will be different for everyone, we need to see your time calculations done by hand to verify that your outputs are correct. These hand calculations can be done in a similar fashion as shown in the 'Examples' under DETAILS or any other way you see fit as long as it is understandable. The answer for this question should match the output of the program in Q2.

For Part 3:

The solution for this can be presented in any form you see fit (written, visual).

FINAL NOTE:

All the non-programming deliverables will be submitted as a PDF. The programming deliverables will be uploaded on a **public** GitHub repository.

- The repository name should be *teamID*-WEC21.
- The repository should have 2 folders labelled 'Part 1' and 'Part 2' which will contain the code for the respective parts.

The PDFs and the repository link must be submitted to this form: https://bit.ly/ProgrammeWEC
Any commits made after the competition ends will not be considered and will lead to negative marking and potential disqualification.

PRESENTATIONS

Teams will create and show a **7 min** presentation and demo for a panel of judges. Order of the presentation and the rooms in which teams present will be determined randomly and will be announced 30 minutes prior to the presentation start time. Teams will be permitted **3 minutes** following the presentation in which judges and the general audience may ask questions. Parts of the presentation should be shared equally between the team members to score full points.

The first-place team will represent the University of Waterloo at the next Ontario Engineering Competition. If the first-place team is unable to attend, the second-place team shall take their place.

MARKING SCHEME

The final team score is based on quantitative evaluation of the solution by five different test cases, qualitative evaluation of use of software engineering principles and best practices, and quality of the team presentation.

Criteria	Poor (<50%)	Good (50 - 75%)	Excellent (75-100%)	Score
Solution Performance				
Part 1: (0 or 5 marks)				
Test case 1				/5
Test case 2				/5
Test case 3				/5
Test case 4				/5
Total:				/20
Part 2:				
Q1: Written solution to compensate for broken button.				/5
Q2: Solution to Q1 implemented in your program correctly.				/10
Q3: Correct hand calculations and output				/5
Total:				/20
Part 3:				
Creativity				/3
Easy to use				/3
Typing Speed				/4
Total:				/10
Software Design and Implementation				
Code Structure	Software structure is haphazard and disorganized. Code is not separated along logical boundaries and may all be contained in a huge monolithic class or function.	Software structure is fairly well-designed. Code is partitioned into smaller functions or classes where appropriate. Attempts are made to design for code modularity.	Software structure shows evidence of planning and design. Architecture follows good design principles and is modular and extensible.	/10
Algorithmic Approach	Solution demonstrates	Solution applies	Solution is elegant and	/10

	little understanding of algorithms/data structures. Algorithm relies on details specific to the particular test cases evaluated. Edge cases are ignored. Solution is inefficient and may rely on a brute force or hard-coded approach.	appropriate data structures and algorithms to solve the problem in a general sense, not just for the test cases. Most edge cases are handled well. Solution takes advantage of problem structure to improve efficiency.	demonstrates a strong grounding in algorithms and data structures. All edge cases are handled robustly. Solution uses advanced algorithm techniques to solve the problem in a highly efficient manner.	
Code Cleanliness	Code is poorly formatted. Use of whitespace is inconsistent. Variable and function names seem confusing or arbitrary. Documentation is non-existent.	Code is mostly consistent and readable. Some areas have poor naming or styling. Comments are lacking for complicated sections that would benefit from explanation.	Code is immaculately formatted throughout and is immediately readable and understandable. Names are chosen appropriately and comments are added when necessary.	/10
Total:				/30
Presentation				
Explanation of design decisions	Solution is presented poorly. Explanation of solution structure and algorithms are missing or unclear. Little or no justification is provided for design decisions.	Solution is explained adequately. Figures and diagrams are used where appropriate. Some design decisions lack justification.	Solution is explained clearly and eloquently. All design decisions are explained and justified.	/10
Communication	Presentation is disorganized and unpractised.	Presentation is mostly organized with only the occasional stumble or misstep.	Presenters are professional, organized, and articulate.	/10
Total:				/20
Bonus:	Up to 10 bonus points may be added at the judges' discretion for exceptional work that goes above and beyond the competition expectations.			/10
Negative Marking				
Late commit (No more than 3 min late)		-6% penalty		
Overtime Presentation		-5 marks per 20 seconds		
Grand Total:				/100

In case of a tie in total marks, the judging panel will confer and select the winning team. Completed marking sheets will not be disclosed to competitors; however, if teams wish to know their strengths and weaknesses for improvement in future competitions, judges and WEC staff may be available after the competition for questions.