CSE 220: Systems Fundamentals I

Stony Brook University
Homework Assignment #0
Spring 2024
Due: Sunday, February 4th, 2024 by 9:00 pm EST

Updates to this Document

- None yet!

Learning Outcomes

1. An ability to implement a simple interactive game in C that uses loops and an array of primitive values.
2. An ability to use VS Code in GitHub Codespaces to implement a C program.
3. An ability to use CMake to build a C program.
4. An ability to use git to manage the source code of a simple C program.
5. An ability to use Codegrade to automatically test a C program.

Assignment Overview

This assignment is intended to give you an introduction to programming in C and to the various tools we will use throughout this semester:

- git for source code management
- CMake for building an executable from C source code
- CodeGrade for testing, grading and collecting your work

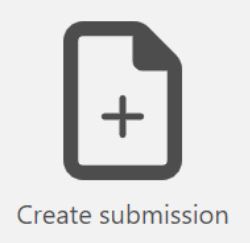
Tchuka Ruma

You will implement the child's solitaire game known as Tchuka Ruma. Read the rules at the linked web page and watch this video demonstration of how it's played, and how it can be solved.

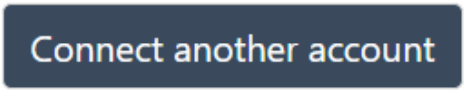
Getting Started from the Template Repository

1. Login to Brightspace and navigate to the Content / Homework Assignments Section. Click the link for Homework #0 to open CodeGrade.

2. In CodeGrade, click on the link for Homework #0 and then look for this button:

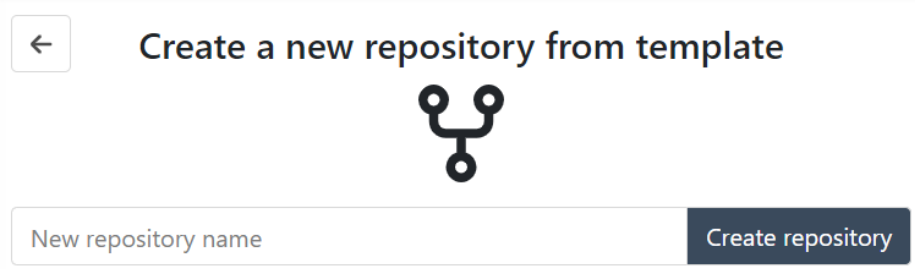


3. Press this button to link your GitHub account to CodeGrade:



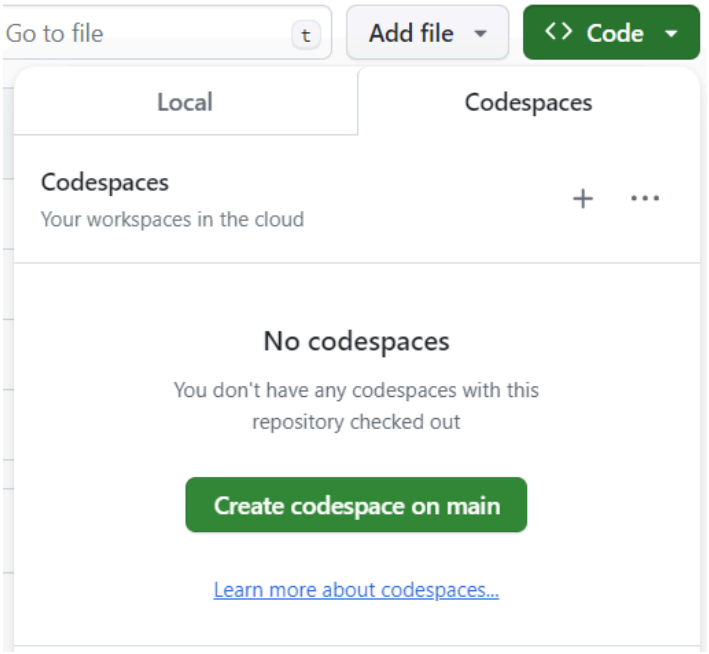
4. Press this button:

5. Sign in to GitHub.
6. Create the repository and give it a sensible name. I recommend “cse220_hw0”.

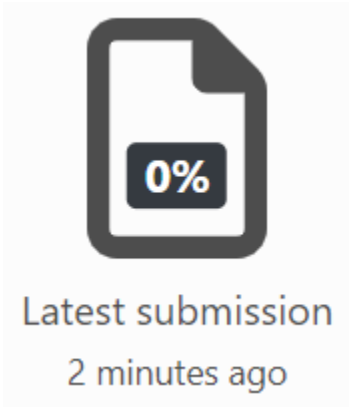


7. In a separate browser window or tab, navigate to your GitHub repositories. Find your cse220_hw0 repo. Use the green **Code** button to create a Codespace so you can start

coding. This process will take a few minutes to complete.



8. To track your success in implementing the program correctly, you must occasionally `git push` your code to GitHub, which will also push it to CodeGrade. In CodeGrade, click this button or one similarly labeled to check the results of the autotests:

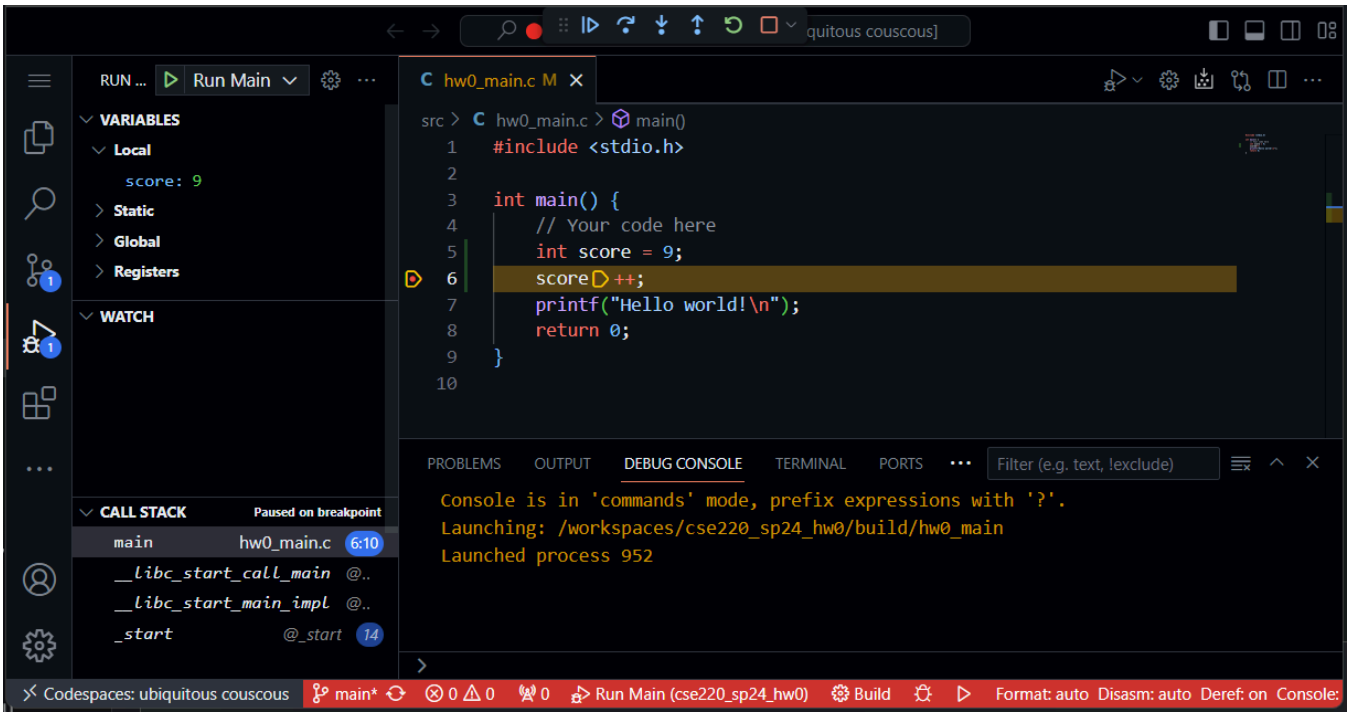


Building and Running Your Code at the Command Line

1. Configure the code to build. This needs to be done only once:
   `cmake -S . -B build`
2. Build the code after changing the code:
   `cmake --build build`
3. Run the program to play Tchuka Ruma:
   `./build/hw0_main`

Building and Debugging Your Code in VS Code

1. To build your code, open the file CMakeLists.txt and hit F7 on the keyboard. If VS Code asks you to select a “kit”, choose the Clang Linux option.
2. To run your code, simply hit F5. Later on, when you make changes to your code and hit F5 again, VS Code will rebuild your code as necessary before running your program.
3. To find bugs, create a breakpoint by clicking in the empty area just to the left of a line of code. A red dot will appear. When you run your code, the debugger will pause execution before executing that line. It will look like the screenshot below. Note the “Variables” panel on the left side of the screen; it’s very useful!



4. This small panel of buttons at the top of the window lets you resume execution (F5), step through your code line-by-line (F10), and terminate execution (shift-F5).

Functionality to Implement

Your implementation must take input from standard input (e.g., using `scanf`) and display all output to standard output (e.g., using `printf`). To facilitate automated grading, input and output must be formatted exactly as given below. An underscore is used in place of a space character in some of the bullet points below.

Print the state of the game

- The state of the game must be displayed as follows: **#_#_#_#_|_#**
- Each pound sign represents the number of counters in each section (labeled 1 through 4 from left to right, with the Ruma at the rightmost position). A vertical bar (“pipe”) separates sections 1-4 from the Ruma.

Print a prompt

- A prompt message must be formatted exactly as follows:
  Choose a section (1-4):_
- Do not print a newline character at the end of the message, only the space character.
- Assuming the player picked a valid section, update and display the state of the game.

Print an error message

- If the user attempts to choose an invalid section number, print the following prompt message to request a new section number:
  Invalid choice. Choose a section (1-4):_
- Do not print a newline character at the end of the message, only the space character.
- After the player has picked a valid section, update and display the state of the game.

Print the “winning” message

- The winning message is: **You won!**
- Print a newline character at the end of the message.

Print the “losing” message

- The losing message is:
  You lost because the last counter fell into section #.
- Substitute the correct section number (1-4) for the pound sign when displaying this message.
- Print a newline character at the end of the message.

Print the “continue sowing seeds” message

- When the last piece on a turn lands in a section that already contains a counter, the player must automatically continue seeding counters. This must happen automatically without user intervention. When this happens, display the following message:
  Last piece landed in section #. Continue sowing seeds!
- Substitute the correct section number (1-4) for the pound sign when displaying this message.
- Print a newline character at the end of the message.
- After sowing seeds, display the updated state of the game.

Sample Gameplay #1

```
2 2 2 2 | 0
Choose a section (1-4): 3
2 2 0 3 | 1
Choose a section (1-4): 2
2 0 1 4 | 1
Last piece landed in section 4. Continue sowing seeds!
3 1 2 0 | 2
Last piece landed in section 3. Continue sowing seeds!
3 1 0 1 | 3
Choose a section (1-4): 2
3 0 1 1 | 3
You lost because the last counter fell into section 3.
```

Sample Gameplay #2

```
2 2 2 2 | 0
Choose a section (1-4): 3
2 2 0 3 | 1
Choose a section (1-4): 4
3 3 0 0 | 2
Last piece landed in section 2. Continue sowing seeds!
3 0 1 1 | 3
Choose a section (1-4): 3
3 0 0 2 | 3
Last piece landed in section 4. Continue sowing seeds!
4 0 0 0 | 4
Last piece landed in section 1. Continue sowing seeds!
0 1 1 1 | 5
Choose a section (1-4): 4
0 1 1 0 | 6
Choose a section (1-4): 2
0 0 2 0 | 6
Last piece landed in section 3. Continue sowing seeds!
0 0 0 1 | 7
Choose a section (1-4): 4
You won!
```

Sample Gameplay #3

```
2 2 2 2 | 0
Choose a section (1-4): 2
2 0 3 3 | 0
```

```
Last piece landed in section 4. Continue sowing seeds!
3 1 3 0 | 1
You lost because the last counter fell into section 2.
```

Sample Gameplay #4

```
2 2 2 2 | 0
Choose a section (1-4): 3
2 2 0 3 | 1
Choose a section (1-4): 3
Invalid choice. Choose a section (1-4): 6
Invalid choice. Choose a section (1-4): 4
3 3 0 0 | 2
Last piece landed in section 2. Continue sowing seeds!
3 0 1 1 | 3
Choose a section (1-4): 1
0 1 2 2 | 3
Last piece landed in section 4. Continue sowing seeds!
1 1 2 0 | 4
You lost because the last counter fell into section 1.
```

Testing & Grading Notes

Every time you `git push` to GitHub, a copy of your code is also submitted and automatically handed in for testing and grading. The output of the test cases can be viewed in CodeGrade. Note that hidden test cases are run against your code as well. Your score is automatically posted to the Brightspace gradebook.

Follow these steps to commit and push your changes to Git:

1. Use the command `git add src/hw0_main.c` to add all your changes.
2. Commit your changes with `git commit -m "Your commit message here"`.
3. Finally, push your committed changes using `git push`.

Note: any evidence of hard-coding test case input or output will automatically result in a score of zero for the assignment.

Generative AI

Generative AI may not be used to complete this assignment.

Academic Honesty Policy

Academic honesty is taken very seriously in this course. By submitting your work for grading you indicate your understanding of, and agreement with, the following Academic Honesty Statement:

1. I understand that representing another person's work as my own is academically dishonest.
2. I understand that copying, even with modifications, a solution from another source (such as the web or another person) as a part of my answer constitutes plagiarism.
3. I understand that sharing parts of my homework solutions (text write-up, schematics, code, electronic or hard-copy) is academic dishonesty and helps others plagiarize my work.
4. I understand that protecting my work from possible plagiarism is my responsibility. I understand the importance of saving my work such that it is visible only to me.
5. I understand that passing information that is relevant to a homework/exam to others in the course (either lecture or even in the future!) for their private use constitutes academic dishonesty. I will only discuss material that I am willing to openly post on the discussion board.
6. I understand that academic dishonesty is treated very seriously in this course. I understand that the instructor will report any incident of academic dishonesty to the University's Academic Judiciary.
7. I understand that the penalty for academic dishonesty might not be immediately administered. For instance, cheating on a homework assignment may be discovered and penalized after the grades for that homework have been recorded.
8. I understand that buying or paying another entity for any code, partial or in its entirety, and submitting it as my own work is considered academic dishonesty.
9. I understand that there are no extenuating circumstances for academic dishonesty.