

Peer Assessments ([https://class.coursera.org/programdesign-002/human\\_grading/](https://class.coursera.org/programdesign-002/human_grading/)) / Project 1 (Editor)

Help ([https://class.coursera.org/programdesign-002/help/peergrading?](https://class.coursera.org/programdesign-002/help/peergrading?url=https%3A%2F%2Fclass.coursera.org%2Fprogramdesign-002%2Fhuman_grading%2Fview%2Fcourses%2F971286%2Fassessments%2F6%2Fsubmissions)

[url=https%3A%2F%2Fclass.coursera.org%2Fprogramdesign-](https://class.coursera.org/programdesign-002%2Fhuman_grading%2Fview%2Fcourses%2F971286%2Fassessments%2F6%2Fsubmissions)



[002%2Fhuman\\_grading%2Fview%2Fcourses%2F971286%2Fassessments%2F6%2Fsubmissions](https://class.coursera.org/programdesign-002%2Fhuman_grading%2Fview%2Fcourses%2F971286%2Fassessments%2F6%2Fsubmissions))

due in 1wk 3d


#### Submission Phase

1. Do assignment ☐ (/programdesign-002/human\_grading/view/courses/971286/assessments/6/submissions)

#### Evaluation Phase

2. Train  (/programdesign-002/human\_grading/view/courses/971286/assessments/6/trainingSets)
3. Evaluate peers  (/programdesign-002/human\_grading/view/courses/971286/assessments/6/peerGradingSets)

#### Results Phase

4. See results  (/programdesign-002/human\_grading/view/courses/971286/assessments/6/results/mine)

☐ In accordance with the Honor Code, I certify that my answers here are my own work, and that I have appropriately acknowledged all external sources (if any) that were used in this work.

[Save draft](#)

[Submit for grading](#)

## Learning Goals

- Be able to comprehend data definitions for compound data.
- Be able to design functions operating on compound data.
- Be able to design world programs on compound data.
- Be able to use the `on-key` option to `big-bang`.
- Be able to use a wish-list to keep track of work remaining to be done in a large program design.

## Introduction

You will complete the design of a simple one line text editor similar to the one you see when you send a text message on your phone or type into the search bar on your browser.

We have started the design of this program, and you will complete it. You will see how the data definitions and wish-lists make it possible for you to finish a program someone else has started.

Once your editor is complete, starting it with `(main (make-editor "abcdef" 0))` should display the following editor:

`abcdef`

The red line is the cursor. In this text editor, you will be able to insert and delete text and move the cursor

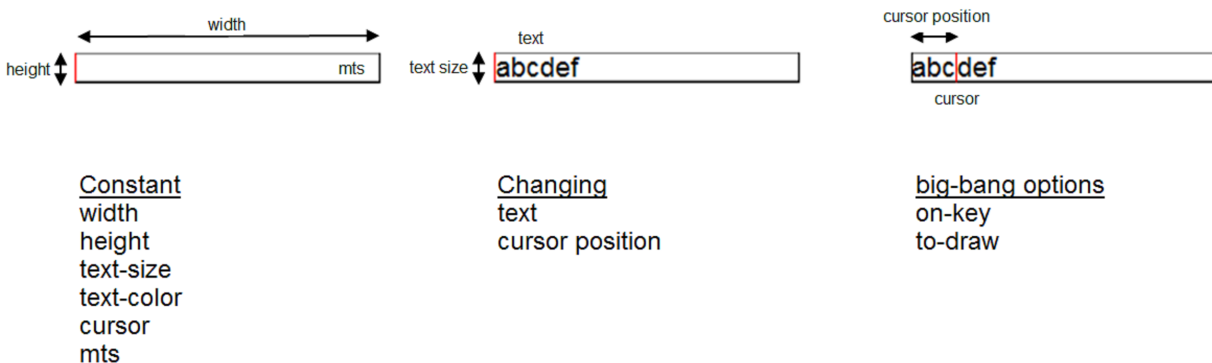
left and right. To help you with the project, we have made a [demo video](https://class.coursera.org/programdesign-002/lecture/163) (<https://class.coursera.org/programdesign-002/lecture/163>) of what the program should do when it's done.

## Main Project

One of the goals for this project is for you to get experience working with a larger program broken into multiple functions. At the very least, we would like you to have each case of `handle-key` call a different function to do that part of the work -- these are often called helper functions. For example, in our solution, the first case of `handle-key` calls a helper function named `move-cursor-left`. So the answer for that `cond` clause is `(move-cursor-left e)`. In our solution, those four helper functions also call two other shared helper functions. We want you to have at least four helper functions for `handle-key`. After that, the number of helper functions you use is up to you - but try to keep to a general guideline of "one task per function". Your experience with this will set us up for more detailed discussions on helper functions after the project.

Note that in the case of `handle-key` we are giving you part of the template, so you only need to complete the `check-expected`s and code the function body.

Before you start programming, you should carefully review the domain analysis provided below.



As you go through this domain analysis, be sure that you understand how we identified all of the constants, changing information and `big-bang` options. Then you should go through the [starter file](https://spark-public.s3.amazonaws.com/programdesign/starters/editor-project-starter.rkt) (<https://spark-public.s3.amazonaws.com/programdesign/starters/editor-project-starter.rkt>) to ensure you understand the constants, data definition, and main function provided there. Use the data definitions and wish-list entries in the program to determine what you need to do to finish it.

If you're unsure about how to complete the functions, one suggestion is to go through the examples and draw images of the examples to help you understand the correspondence between Editor and what appears on the screen. In other words **follow the recipe!**

When you are done, the editor should implement at least the following functionality:

- The left arrow key should move the cursor left (unless already at left end of text).
- The right arrow key should move the cursor right (unless already at right end of text).
- The backspace key should delete the character before the cursor (if there is one).
- Key events of length 1 are normal characters, these should be inserted at the cursor position.

**Note:** You are not required to handle the DEL key on a PC. You only need to handle the `"\b"` key, which is labeled Delete on a Mac and Backspace on a PC. You only need to handle the Delete/Backspace key found on the main keyboard.

Don't worry about making a fully fledged editor! Don't worry about cases like the box getting too full. The goal is to have a simple design that works properly.

## Downloads

1. [starter file](https://spark-public.s3.amazonaws.com/programdesign/starters/editor-project-starter.rkt) (<https://spark-public.s3.amazonaws.com/programdesign/starters/editor-project-starter.rkt>)

2. [demo video \(https://class.coursera.org/programdesign-002/lecture/163\)](https://class.coursera.org/programdesign-002/lecture/163)
3. [subtitles \(https://class.coursera.org/programdesign-002/lecture/subtitles?q=163\\_en&format=srt\)](https://class.coursera.org/programdesign-002/lecture/subtitles?q=163_en&format=srt)

## Submission

Upload your project below as a single .rkt file. If you don't submit your own assignment before the deadline, your work will not be graded by your peers, and you will not receive a grade. Make sure that your work is actually submitted, and not just saved.

<b>B</b>	<i>I</i>			Link	<code>	<HTML>

☐ Default Editor

[Attach a file](#)

### Overall evaluation/feedback

**Note:** this section can only be filled out during the evaluation phase.

Download the submitted .rkt file. All of your grading will be based on the .rkt file. If there is more than one file in the submission box, you only need to grade the first one.

#### Important

Before you run the file, look over the file to make sure there are no new require declarations anywhere in the file. This is important to make sure there is no malware in the file. There should only be `(require 2htdp/image)` and `(require 2htdp/universe)` at the beginning of the file. You should use search to make sure there are no others anywhere else in the file. If you find a single extra require stop grading, the entire grade for the project is 0.

#### Program is Well-Formed and Tests Pass (Maximum 4 Points)

A proper solution should be well-formed so that all the tests at least run. A program with failing tests is much easier for another programmer to work on than a program that is not even well-formed.

**1 point** if there is no call to main at the bottom of the program

**1 point** if the tests run

**1 point** if no tests are commented out

**1 point** if all tests pass

### Correctness (Maximum 4 Points)

Run the program. For nearly all submissions it will be sufficient to simply type `(main (make-editor "" 0))`. But if someone has changed the data definitions for some reason you might need to do something else.

**1 point** if you can insert characters when you type a character

**1 point** if you can delete characters when you press the backspace key

**1 point** if the cursor moves left when you press the left arrow key

**1 point** if the cursor moves right when you press the right arrow key

Note: The backspace key refers to the `"\b"` key, which is labeled Delete on a Mac and Backspace on a PC.

### Following Recipes (Maximum 10 Points)

**2 points** if `handle-key` has at least 5 `check-expect` s (there must be at least one for each of the four cases that have behavior and one for an ignored key like shift)

**2 points** if the helper function that moves the cursor left (the function might be called `cursor-left` or a similar name) has adequate `check-expect` s. There must be at least 2 `check-expect` s (one for the case when the cursor is at the far left and one for the case when the cursor is anywhere else). If there is no helper that does this, then award **0 points**.

**2 points** if the helper function that deletes the character before the cursor has the correct signature. The signature should be `Editor -> Editor`. If there is no helper that does this, then award **0 points**.

**2 points** if all functions not already in the starter have an appropriate purpose. If the only functions written were defined in the starter, then award **0 points**.

**2 points** if all functions not already in the starter have an appropriate signature. If the only functions written were defined in the starter, then award **0 points**.

### Using Helpers (Maximum 2 Points)

**2 points** if each case of `handle-key` calls a helper function (one for each of the four cases that have behavior)

### Coding Style (Maximum 4 Points)

The grade for Coding Style should be based only on code that has been added to the starter. If no new code has been added to the starter, then award **0 points** for Coding Style.

**2 points** if there are no commented out `check-expect`s or commented out functions in the program. Stubs, templates, and brief explanatory test comments are permitted but not required.

**2 points** if all the functions are indented and named properly. The code should be indented using the conventions seen in our examples. Function names should make sense with respect to the problem and should be hyphenated and not capitalized. See the Style Rules (<https://class.coursera.org/programdesign-002/wiki/view?page=StyleRules>) page for more information.

### General Feedback (0 Points)

Please make one or two comments about the program. Was it well structured? Easy to read? Did it have a particularly interesting solution to one part of the problem? Use this space to give the student free-form commentary about their design. Remember, they worked very hard on it, so be positive, but also make comments you feel might be helpful to them in terms of improving their work. Please restrict your comments to the scope of weeks 1-4 of this course.

**Privacy choice:** (read [privacy policy \(//class.coursera.org/programdesign-002/wiki/view?page=PrivacyPolicy\)](https://class.coursera.org/programdesign-002/wiki/view?page=PrivacyPolicy))

☒ I'd love to help out the course with my work. The staff may share my submission with my classmates and other people who are curious, for things like peer assessment training, lecture slides, and publications.

---

☐ In accordance with the Honor Code, I certify that my answers here are my own work, and that I have appropriately acknowledged all external sources (if any) that were used in this work.

[Save draft](#)

[Submit for grading](#)