# COMP6080 Final Exam

For this exam you are provided with this public repostory (`exam-spec`) that all students have access to. This repository contains the questions being asked. You will then also have your own [personal exam repository](https://cgi.cse.unsw.edu.au/~cs6080/redirect/?path=COMP6080/24T1/students/\_/exam) where you actually complete the work that will be submitted. The personal exam repo is where you actually commit and push your code.

If you are seeking information not provided in this file, check the [COMP6080 Exam Brief page](https://cs6080.web.cse.unsw.edu.au/24T1/assessments/exam). If your question is still unanswered, follow the "Communicating with teaching staff" instructions.

## WARNING

This applies to all students completing the COMP6080 24T1 exam on [TODO].

\* This exam is an individual assessment. Any attempt to communicate with other people (both other students in this course and outside persons) about the contents of this exam will be treated as academic misconduct and may result in you failing this course. This applies to everyone during the exam time, as well as any communication between the end of your exam and Wednesday the 12th of May at 1pm. To avoid any doubt about your behaviour during the exam, cease all communication with other students for that time.

\* From 1pm on Wednesday the 12th of May onward, you are only allowed to discuss the exam with students who have themselves also completed the exam, and it's your responsibility to check if they have.

\* Your zpass should not be disclosed to any other person. If you have disclosed your zpass, you should change it immediately.

\* Do not place your exam work in any location accessible to any other person. This includes services such as Dropbox and Github.

\* If another student in the course makes any sort of contact with you during the exam, or you’re aware of any instances of other students breaching the conditions above, you are required to email cs6080@cse.unsw.edu.au with details of the interaction.

## Change Log

N/A

## 1. The Exam

### 1.1 Overview

You are to build a small single page application (with either ReactJS or Vanilla JS) that has three basic interactive games, along with a dashboard acting as a homepage.

### 1.2. Getting Started

Please clone your [personal exam repository](https://cgi.cse.unsw.edu.au/~cs6080/redirect/?path=COMP6080/24T1/students/\_/exam).

Run `yarn install` to install all relevant dependencies to start.

Run `yarn start` to start your ReactJS app.

You are welcome to install any dependencies on top of ReactJS that you would like using `yarn add [dependency]`.

Please note: If you prefer to complete the exam with VanillaJS, then you can simply remove all files we've provided and start from an empty repo.

There is no backend in this application. The entire state you manage should be done in localstorage or another form of persistent browser storage.

### 1.2. Features

The numbers in brackets next to any items (e.g. "Make the page big (1)") describe the number of marks associated with that piece of functionality.

#### 1.2.0. Document, sidebar, and Footer (11 marks)

\* The overall document/page should have a `margin` of `0px`. (0.5)

\* All screens shall have a sidebar bar that is:

\* `100px` wide. (0.5)

\* On the right hand side of the page. (0.5)

\* Has a height that spans the full height of the viewport. (0.5)

\* Has property `position: fixed` such that no matter where the user scrolls the sidebar is visible. (0.5)

\* Has a background colour of `#eee`. (0.5)

\* Contains a logo (any random image from the internet) that:

\* At the top of the sidebar, centre aligned. (0.5)

\* Has a margin of `15px` on top and bottom. (0.5)

\* Has a size `50px` x `50px`. (0.5)

\* Contains 4 lines of text that are links: (1)

\* Home (linking to /home)

\* Operations (linking to /operations)

\* Memory (linking to /memory)

\* Space (linking to /space)

\* If the screen width drops below `1400px`, then:

\* The linktext, instead of "Home", "Operations", "Memory", "Space" becomes "H", "Op", "Me", "S" (1)

\* The width of the side bar drops to `60px` (0.5)

\* If the screen width drops below `800px`, then:

\* The linktext, instead of "Home", "Operations", "Memory", "Space" becomes "H", "Op", "Me", "S" (1)

\* The width of the side bar drops to `30px` (0.5)

\* The logo disappears (0.5)

\* All screens shall have a footer bar that:

\* Has a height of 50px. (0.5)

\* At the bottom of the page at all time (0.5)

\* Has a width that spans the full width of the viewport, minus the sidebar. (0.5)

\* Has a background colour of `#999` (0.5)

\* Contains no content

\* The entirety of the space on the page that isn't used by the sidebar and footer is referred to as the `main body`, and screens from `1.2.1`, `1.2.2`, `1.2.3`, and `1.2.4` should occupy that full space, unless otherwise specified.

#### 1.2.1. Dashboard (11 marks)

\* This screen exists on route `/home` and contains the sidebar and footer from `1.2.0`. (0.5)

\* This screen shall contain two lines of text that are center aligned:

\* The first line is the text `Please choose an option from the sidebar.` The text is colour `blue` and font size `2em`. (1)

\* The second line is the content `Games left to win: X (reset)`. (1)

\* `X` should be initially set as the number found in the JSON object (format `{"score":5}`) at this URL: [https://cs6080.web.cse.unsw.edu.au/raw/data/score.json](https://cs6080.web.cse.unsw.edu.au/raw/data/score.json). (2)

\* Each time the player wins any of the 3 games, this number should go down. (1)

\* The value of `X` shall persist between loads by making use of a form of browser storage (e.g. `localstorage`). (2)

\* Once the value reaches 0, an alert is shown saying 'Congratulations!' and the value is reset. This alert only has to appear on navigation to `/home`. (1)

\* This value should be stored in local storage so that after first load, and prior to a reset, it stays persistent (1.5).

\* When `(reset)` (a button) is clicked, the counter `X` will be reset to the original value. (1)

#### 1.2.2. Game 1 - Operation Blanks (18 marks)

\* This page exists on route `/operations` and contains the sidebar and footer from `1.2.0`. (0.5)

\* This main page consists of a primary box that is:

\* `100px` in height (0.5)

\* `100%` in width (0.5)

\* Vertically spaced equally between the top and bottom of main page (2)

\* Has a background of `rgb(200,255,255)` (0.5)

\* Contains a row of 5 equally spaced sections within this box (1)

\* The first section is \*\*Input number 1\*\*, a single number that is centre aligned (horizontally and vertically) in the box (1)

\* The second section is made up of 4 buttons. These buttons can be laid out in the section however you want. The buttons should contain the text `+`, `-`, `x`, and `÷` respectively (to represent the 4 operations). (2)

\* The third section is \*\*Input number 2\*\*, a single number that is centre aligned (horizontally and vertically) in the box (0.5)

\* The fourth section is just the character `=` which is centre aligned (horizontally and vertically) in the box (0.5)

\* The third section is \*\*Output number\*\*, a single number that is centre aligned (horizontally and vertically) in the box (0.5)

\* When the game starts, it populates \*\*Input number 1\*\*, \*\*Input number 2\*\*, and \*\*Output number\*\* with 3 different numbers. You can generate these numbers randomly yourself, or you can use numbers provided to you in the Javascript snippet below. (2)

\* Once the numbers are populated, the user must then press one of the 4 buttons on the screen.

\* If the button pressed successfully completes the formula such that it is valid (e.g. `2 + 2 = 4`) then the user receives an alert saying that they have won the game (3)

\* If the button pressed does not complete a formula successfully, then the user receives an alert saying their answer was incorrect. The user then has unlimited attempts to get it right. (1)

\* Every time the game is won, a new set of numbers must appear. After 5 unique sets have been displayed, it's OK to cycle back through the previous numbers. It's also OK to cycle back to the initial set after each reload. (2.5)

```javascript

const numbers = [

[1,2,2],

[3,6,-3],

[8,3,11],

[9,8,17],

[5,4,9],

]

```

[A short video demonstration of this game can be found here](https://www.youtube.com/watch?v=7hm0CvjArws). Please note - do not copy what you see in this video. There are parts of this video that may not comply with the specification. This is purely here to give you a very general sense of the behaviour in case you are struggling to visualise it.

#### 1.2.3. Game 2 - Flashing Memory Game (22 marks)

\* This page exists on route `/memory` and contains the sidebar and footer from `1.2.0`. (0.5)

\* The top half of the main body should consist of 4 boxes that have uniformly distributed width, and a height that fills half the main body. (2)

\* Each box itself should be a clickable button area, where the 4 boxes contain the innerText of `A`, `B`, `C`, and `D` respectively (2)

\* The bottom half of the main body contains a box (called the "instruction box") that:

\* Is `20px` x `20px` in size (0.5)

\* Is center aligned (both horizontally and vertically) within the bottom half (1)

\* Has a background of `#cccccc` and no border (0.5)

\* When the page mounts, the game begins. (0.5)

\* When the game begins, it starts the first "round" where that round consists of the first iteration (X = 1) as follows:

\* 1) All of the 4 buttons in the top half are disabled (1)

\* 2) A random character (out of `A`, `B`, `C`, and `D` appears in the instruction box) (0.5)

\* 3) This random character appears for exactly 1 second, before disappearing again (1)

\* 4) Steps 2 and 3 are repeated until X random characters are shown (5)

\* 5) The 4 buttons in the top half are re-enabled (0.5)

\* 6) The user is then allowed to click X buttons in the top half. If the correct values are entered in the right order, the user moves to the next iteration where X is 1 value higher. If they are entered incorrectly, the user loses the game. (5)

\* Once the user successfully completes an iteration where X = 5, the user receives an alert that says they have won the game (1)

\* If the user loses the game, an alert popups up informing them of this and then the game starts again (1)

[A short video demonstration of this game can be found here](https://www.youtube.com/watch?v=3akIxDHura4). Please note - do not copy what you see in this video. There are parts of this video that may not comply with the specification. This is purely here to give you a very general sense of the behaviour in case you are struggling to visualise it.

#### 1.2.4. Game 3 - Space Invaders (19 marks)

This page is a simplified implementation of the game [space invaders](https://en.wikipedia.org/wiki/Space\_Invaders). For this page, you do not need to worry about mobile responsiveness as we will assume a fixed size.

\* This page exists on route `/space` and contains the sidebar and footer from `1.2.0`. (0.5)

\* This page will consist of a game window that the game is played in. This window will have a `1px` black border and its size will be `500px` by `500px` (0.5)

\* When the component is mounted, the game should begin immediately (0.5).

\* When the game begins:

\* A small `10px` x `10px` red square (the shooter) should appear in the bottom left corner (1)

\* A series of `20px` x `20px` black squares (the objects to shoot) should appear up the top of the game window. (1)

\* They should have margins of `15px` each (0.5)

\* They should go all the way along the screen horizontally. This means there should be 10 squares in a row (see demo video for examples). (4)

\* There should be 2 rows of these black squares (see demo video for examples). (1)

\* Anytime whilst the game is being played:

\* If the left or right keyboard arrows are pressed the red square (the shooter) should move `1px` to the left or the right respectively. (2)

\* The shooter should be unable to move beyond the bounds of the game window (2)

\* If the space bar keyboard key is pressed and any black squares overlap vertically with the shooter (red square), the black square should disappear. (5)

\* When the game finishes (i.e. all black squares destroyed) the user should receive an alert saying that they have won. The game should then restart (1).

[A short video demonstration of this game can be found here](https://www.youtube.com/watch?v=dmrX0CPlIOE). Please note - do not copy what you see in this video. There are parts of this video that may not comply with the specification. This is purely here to give you a very general sense of the behaviour in case you are struggling to visualise it.

### 1.3. Other notes

\* If we don't specify a constraint, then you have discretion as to what to do, assuming it still ensures that your application is usable and accessible.

\* If a CSS property constraint is not specified (e.g. font size) then you are free to use whatever is reasonable and usable.

\* While we don't specify many requirements around usability and accessibility, you should take initiative to make your work both usable and accessible to gain the marks in that area.

\* You should ensure that your programs have been tested on one of the following two browsers:

\* Locally, Google Chrome

\* On CSE machines, Chromium

## 2. Marking Criteria

For each of sections, marks will be awarded according to the following criteria:

\* 80%: Providing the features and functionality required in at least one of desktop, tablet, or mobile views.

\* 20%: Ensuring responsiveness on desktop, tablet, and mobile

\* Desktop testing will be done on a `1800px` x `800px` viewport size

\* Tablet testing will be done on a `1200px` x `500px` viewport size

\* Mobile testing will be done on a `600px` x `500px` viewport size

## 3. Submission

At the end of your specified exam time, we will automatically collect the code on your `master` branch's HEAD (i.e. latest commit).

Please note: If you develop locally ensure you check that your code works on the CSE servers. Failure to do so could result in a fail mark in the exam.

## 4. Originality of Work

The work you submit must be your own work. Submission of work partially or completely derived from any other person or jointly written with any other person is not permitted.

The penalties for such an offence may include negative marks, automatic failure of the course and possibly other academic discipline. Assignment submissions will be examined both automatically and manually for such submissions.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct.

Do not provide or show your assignment work to any other person — apart from the teaching staff of COMP6080.

If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted, you may be penalized, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.

Note you will not be penalized if your work has the potential to be taken without your consent or

knowledge.