Overview

For this assignment, you will be writing a program to play the card game *Intensity*.

The Rules of Intensity

Intensity is an unusual card game which is thought to have originated in post-WWII buffalo hunting camps in Australia's Northern territory. The name in**tens**ity is thought to derive from numbers in groups of ten being important.

Cards

It is played with a deck of 40 cards.

The cards are numbered 10 to 49.

No two cards have the same number.

11 cards have special significance. They are the cards numbered 30 to 39, usually called the *calves*, and the card numbered 47, usually called the *buffalo*.

Game Play

Intensity is played by four players sitting around a circular table. These players are numbered clockwise: 0, 1, 2 and 3. The deck is divided randomly among the 4 players. So each player starts with a **hand** of 10 cards.

Stage 1: Discarding

In the first stage of play, each player selects three cards from their hand to be passed to the player on their left (clockwise). They, of course, then receive three cards from the player on their right. So player 0 passes to player 1, player 1 to player 2, player 2 to player 3 and player 3 to player 0.

The four players simultaneously select the three cards to be passed on. Each player must pass on 3 cards, before receiving the three cards from the player on their right. Hence, you must select the three cards you will pass to the player on your left before you know which three cards you have been given by the player on your right.

Stage 2: Playing Cards

The second stage of play consists of a series of 10 **rounds**. The goal of the game is to avoid penalty points during these ten rounds.

A **round** begins with a designated player selecting a card from their hand and playing it. Then proceeding clockwise, the other players, in turn, each select a card from their hand and play it.

Hence for the first round, each player will have ten cards in their hand to select from, for the second round they will have nine cards in their hand and so on. For the tenth and last round, each player will have only one card and hence, no choice in the card they play.

Playing a Card

The first digit of the first card played in a round is important. The subsequent players **must** play a card of the same first digit as the first card if possible.

For example, if the first card played in a round is the **17**, the other 3 players must play a card with a first digit of '**1**' if they have one in their hand.

If one of the players does not have a card with a first digit of '**1**' in their hand they may play a card of another first digit. The following players are still required to play a card with a first digit of '**1**' card if they have one.

Winning a Round

A round is won by the person who plays the card of the same first digit as the first card in the round with the largest number. For example, if the first card in a round is the **24**, the card with the largest number with first digit '2' will win the round. If no other card with the first digit '2' played the **24** wins.

Player 0 plays the first card of the first round. Subsequently the winner of each round plays the first card of the next round.

If, for example, player 2 won the last round, then player 2 would play the first card of the next round and then players 3, 0 and 1 would play in that order.

Buffalo and Calves

One restriction to the above rules is that a player is not allowed to play a *calf* (a card in the range 30..39) as the first card in a round unless a *calf* has been played in any of the previous rounds or they have only *calves* in their hand. This restriction does not apply to playing a *calf* as the second, third or fourth card in a round.

In other words, if no cards in the range 30..39 have been previously played, a card in the range 30..39 can not be played as the first card of a round unless the player has no choice.

There is no restriction on when the *buffalo* can be played as the first card.

Penalty Points

The aim of *Intensity* is to avoid penalty points. A player scores penalty points if certain cards occur in rounds they have won. Each *calf* is worth one penalty point and the *buffalo* (47) is worth 7 penalty points.

Hence, generally players try to avoid winning rounds which contain *calves* and they particularly try to avoid winning rounds which contain the *buffalo*.

If a round doesn't contain a *calf* or the *buffalo* it doesn't affect the score.

When all 10 rounds are played the penalty points are calculated for each player. The winner of the game is the player with the fewest penalty points.

If a player attempts to play an illegal card they receive five penalty points. The referee will instead select a legal card from their hand at random. This same applies to discards.

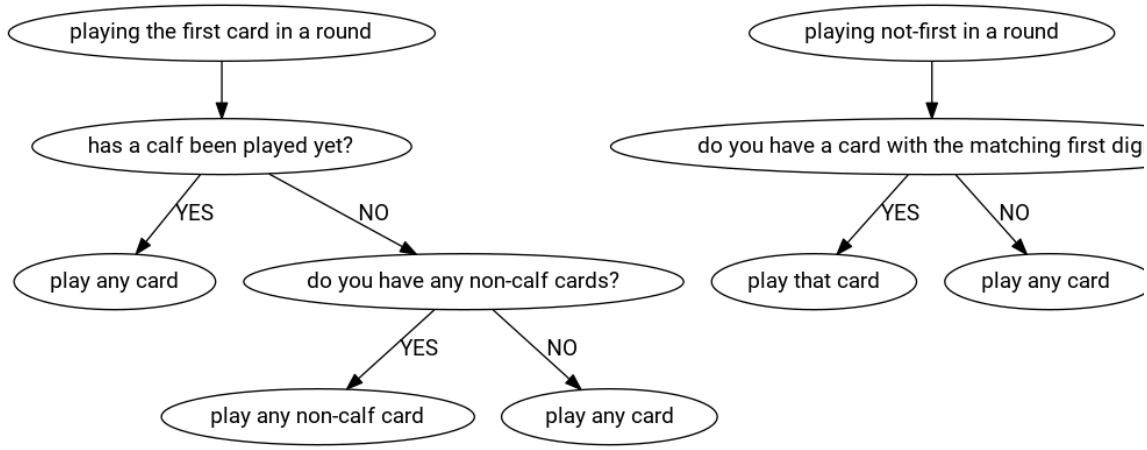
Help + Resources

Rules Video

To fully understand these rules, it might be worth watching this tutor-made video which runs through an example game of intensity.

Valid Moves Diagram

The following diagram visualises the rules for what cards are valid to play at any given point during the game.



Implementation

Intensity in C

Your task in this assignment is to write a C program which plays *Intensity* well.

Your program will need to take in relevant information about the state of the game, use that information to determine which move it should make, and then make that move.

Your program will be given this information as a series of integers on standard input, which describe the state of the game and indicate what it is being asked to do.

There are three sub-tasks that your program will be asked to do, over the course of a game of *Intensity*:

1. Print your player name
2. Select the cards to discard
3. Select which card to play

Your program will be run multiple times during the game: once for each "decision" that it must make.

For example, your program will be run once to select which 3 cards it wishes to pass on. This occurs once at the start of the hand. When it is your turn to play, your program will be asked which card it wishes to play. This will happen 10 times during the playing of a hand.

Intensity in C - Printing Your Player Name

If the first integer on standard input is **0**, your program is being asked to print the name of your player. Your program will be run once in this way at the start of the game.

Your program should print your chosen name as a single line. The name can be anything you like - but please keep it tasteful - definitely nothing offensive or obscene. There is a limit of 32 characters.

A suffix will be added if needed to make your name unique.

For example:

```
$ dcc intensity.c -o intensity
$ ./intensity
0
Dennis Ritchie
$
```

Intensity in C - Selecting Discards

If the first integer on standard input is **1**, your program is being asked to select the 3 cards it wishes to discard. Your program will be run once in this way at the start of the game.

The first integer (1) will be followed by 10 integers which are the cards in your hand. These will all be numbers in the range 10..49. They will be in sorted (increasing) order.

You should read these 10 integers into an array, and then choose which 3 you wish to discard.

You program should then indicate the 3 cards it wishes to discard by printing a single line containing 3 integers. Each integer should be one of the cards in your hand and hence must be in the range 10..49.

For example:

```
$ dcc intensity.c -o intensity
$ ./intensity
1
13 19 24 25 29 32 39 44 46 49
29 39 49
$
```

In the above example the player has these cards in its hand: **[13, 19, 24, 25, 29, 32, 39, 44, 46, 49]**
It chooses to discard this 3 cards: **[29, 39, 49]**.

Intensity in C - Playing A Card

If the first integer on standard input is **2**, your program is being asked to play a card. Your program will be run ten times in this way during a game.

The first integer (2) will be followed by 3 more integers:

1. how many cards are in your hand. This will be in the range 1..10.
2. how many cards have already been played this round . This will be in the range 0..3.
3. your table position. This will be in the range 0..3.

Cards In Your Hand

Next will be the cards in your hand. This will be 1..10 integers in the range 10..49 They will be in sorted (increasing) order.
You should use scanf to read these into an array. Note you have already read a value telling you how many cards are in your hand so you know how many values to read

Cards Played So Far This Round

Next will the cards that have been played so far this round. This will be 0..3 integers in the range 10..49 They will be in the order they were played - the first will be the first card played in this round.

You should also use scanf read to these into an array. Note you have already read a value telling you how many cards have been played this round so know how many values to read.

Cards Played In Previous Rounds

Next will be all the cards played in previous rounds. This will be 0..36 integers in the range 10..49. They will be in the order they were played - the first will be the first card played in the game.

Again you should also use scanf read to these into an array. You can calculate how many values to expect from the number of cards left in your hand. For example if there are 7 cards in your hand there have been 3 previous rounds so there have 3*4 = 12 cards played in previous round

You can work out which player played each of the previous cards, because you know the player in table position 0 played first in the first round, and you know the player who won the first round played first in the second round, and so on.

Cards Your Player Discarded

Next will be the cards your player discarded at the start of the game. This will be 3 integers in the range 10..49. They will be in sorted (increasing) order. These will only be of interest if you are implementing a clever strategy.

Cards Your Player Received

Last will be the cards your player received in the discard round at the start of the game. This will be 3 integers in the range 10..49 They will be in sorted (increasing) order. These will only be of interest if you are implementing a clever strategy.

Playing A Card

Your program should then indicate the card it wishes to play by printing a single line containing an integer. This integer should be one of the cards in your hand and hence must be in the range 10..49.

For example:

```
$ dcc intensity.c -o intensity
$ ./intensity
2
9 2 0
11 15 18 23 33 34 47 48 49
24 26
16 13 19 12
37 38 39
34 47 48
23
$
```

In the above example the player has 9 cards in its hand: **[11, 15, 18, 23, 33, 34, 47, 48, 49]**
Two cards played have been played this round: **[24, 26]**.

There has been 1 previous round in which the cards played were: **[16, 13, 19, 12]**

The player is sitting at table position **0**.

In the discard round this player discarded: **[37, 38, 39]** and received **[34, 47, 48]**

The program played the only card that is legal to play in this situation, the **23**.

To fully understand these rules, it might be worth watching this tutor-made video which answers some common questions students have about inputs.

Intensity in C - Unit tests

If the first integer on standard input is 3, your program is being asked to run the unit tests you have added to **run_unit_tests**

Playing Intensity

The Intensity Referee

You do not have to write C to deal the cards, keep score or manage the playing of the game.

You are given a referee program which does this.

It takes as arguments, the names of 1 or more C programs which play the game. It compiles the programs and runs them giving them input as described above.

If there are less than 4 programs, the referee provides players (named **Lulu**, **Morgan**, **Amy** and **Rat**) who make random legal plays.

Run it like this:

```
$ 1511 intensity_referee intensity.c
Version: 0.2
dcc -o intensity intensity_random.c
dcc --valgrind -o intensity-valgrind i

Deal:
Table position 0: COMP1511 student: [1
Table position 1: Lulu             : [1
Table position 2: Amy              : [2
Table position 3: Morgan           : [1

...
```

Interactive Intensity

An interactive **Intensity** player is available. It allows the (human) user to interactively choose the plays for one player.
Playing against Lulu, Amy and Morgan is not challenging but it may improve your understanding of the game.

Run it like this:

```
$ 1511 intensity_referee -i
Version: 0.2

Discards:

Interactive Player must choose discard
Hand: [11, 12, 18, 19, 26, 28, 32, 33,
Enter cards to discard: 32 33 35

Round 0: Morgan              plays the
Round 0: Amy                 plays the
Round 0: Lulu                plays the
```

```
Interactive Player turn to play.
Hand: [11, 12, 18, 19, 21, 22, 26, 28,
Enter card to play:
```

Example Games

An example Intensity game annotated with explanatory comments can be found here These may help you understand the rules of **Intensity**.

Intensity Tournament

An Intensity tournament will be run starting Monday. Submit your work using give to take part.

In an *Intensity* tournament many *Intensity* games are played. The winner of each game receives 3 tournament points, the second placed player receives 2 tournament points and the third placed player receives 1 tournament point. The winner of the tournament is player with the most tournament points.

Hence in tournament play it is the finishing position in each game that is important, rather than the actual number of penalty points This is a fine distinction which will only affect players attempting sophisticated strategies.

In each round of a tournament, each player takes part in one four player game of Intensity. The four player games are formed so that players with similar numbers of tournament points play each other. Hence players of similar strength tend to play each other.

It may be necessary to use system players in one game, if there number of tournament participants is not divisible by four. This will be in the weakest game.

Code

Initial Code

Here is some code to start the assignment.

Read this file carefully. It is strongly recommended you use this file to start the assignment.

Add your code to this file in the indicated places.

Do not change other parts of the file.

Start by adding code to create a player which makes only a legal play.

When you have a program that always makes a legal play, think of a **simple** strategy to choose good cards to discard at the start of the hand. Implement this strategy.

Next think of **simple** strategies to choose a good (and legal!) card to play and implement them. You will need to create separate functions to make your code readable.

When you add functions try to add some good unit tests to **run_unit_tests**. You may need to declare and initialize arrays in **run_unit_tests** to do this.

Clarifications/Assumptions/Restrictions

You should follow discussion about the assignment in the class forums. Questions about the assignment should be posted there so all students can see the answer.

Your code must be submitted in a single C file named **intensity.c**.

Your submitted code must be C only. You may not submit code in other languages. You may not use *system* or other C functions to run external programs.

You may call functions from the standard C libraries (e.g. the functions from `stdio.h`, `stdlib.h`, `string.h`) and the maths library (`math.h`).

Your **program** must take at most 30 seconds to return a play or discards on a CSE lab computer when compiled with dcc --valgrind.

You are required to submit intermediate versions of your assignment (see below).

There is no way for your program to pass information from one play to the next.

Your program is not permitted to create files.

Assessment

Attribution of Work

This is an individual assignment. The work you submit must be your own work and only your work apart from exceptions below. Joint work is not permitted.
You may use small amounts (< 10 lines) of general purpose code (not specific to the assignment) obtained from site such as Stack Overflow or other publically available resources. You should attribute clearly the source of this code in a comment with it.

You are not permitted to request help with the assignment apart from in the course forum, help sessions or from course lecturers or tutors.

Do not provide or show your assignment work to any other person (including by posting it on the forum) apart from the teaching staff of COMP1511.

The work you submit must otherwise be entirely 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. 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 is taken without your consent or knowledge.

Submission of Work

You are required to submit intermediate versions of your assignment. Every time you work on the assignment and make some progress you should copy your work to your CSE account and submit it using the give command below.

It is fine if intermediate versions do not compile or otherwise fail submission tests.

Only the final submitted version of your assignment will be marked.

All these intermediate versions of your work will be placed in a git repo and made available to you via a web interface at this URL, replace *z5555555* with your zpass.
`https://gitlab.cse.unsw.edu.au/z5555555/18s1-comp1511-ass2/commits/master`

This will allow you to retrieve earlier versions of your code if needed.

You submit your work like this:

```
$ give cs1511 ass2 intensity.c
```

Assessment

This assignment will contribute 12% to your final mark.

80% of the marks for this assignment will be based on the performance of your *Intensity* program, i.e. how well it plays *Intensity*.

20% of the marks for assignment 2 will come from hand marking of the readability of the C you have written. These marks will be awarded on the basis of clarity, commenting, elegance and style. In other words, your tutor will assess how easy it is for a human to read and understand your program.

Tutors will also examine your testing strategy and unit tests. Describe your testing strategy above the **run_unit_tests** function. Here is an indicative marking scheme.

HD (85-100%)	successfully implements good strategies, performs well in Intensity tournament, well-explained beautiful code
DN (75-85%)	successfully implements some strategy, competes successfully against Lulu, Morgan and Amy, readable

	code
60%	always makes a legal play, code is mostly readable
40-50%	serious attempt on assignment
-70%	Knowingly providing your work to anyone and it is subsequently submitted (by anyone).
-70%	Submitting any other person's work. This includes joint work.
0 FL for COMP1511	Paying another person to complete work. Submitting another person's work without their consent.

The lecturer may vary the assessment scheme after inspecting the assignment submissions but it will remain broadly similar to the description above.

Due Date

This assignment is tentatively due Monday 14 May 23:59:59
If your assignment is submitted after this date, each hour it is late reduces the maximum mark it can achieve by 2%. For example if an assignment worth 74% was submitted 10 hours late, the late submission would have no effect. If the same assignment was submitted 15 hours late it would be awarded 70%, the maximum mark it can achieve at that time.