Race Condition™

Game Instructions, v 2.0

Race Condition™ is a card game in which two competing players take turns modifying the same computer program until one of them achieves his or her objective: setting the variable ***x*** to +5 (player "Positive"), or -5 (player "Negative"). This is a complicated game! Please read this booklet!

Types of Cards

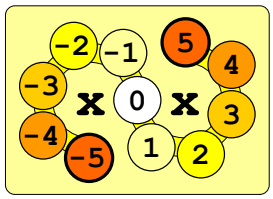
The game's 54 cards fall into 4 categories:

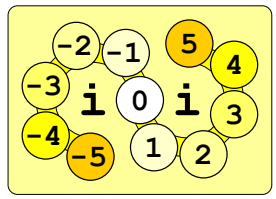
Reference Cards (White)

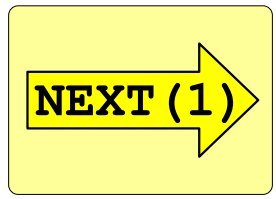
The 2 reference cards have a white background, and simply augment these instructions.

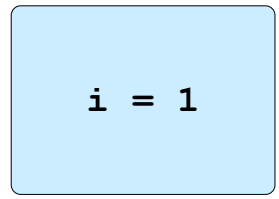
Start Cards (Brown Backs)

The four Start Cards have an S in the upper left corner, have a yellow or blue background, and have a brown back so that they can be easily distinguished from other cards. The Start Cards appear below.





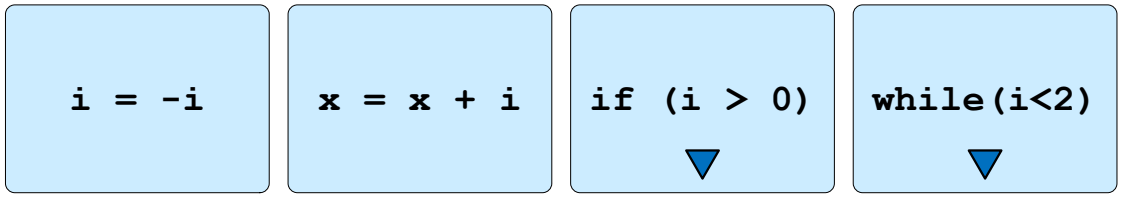




Instruction Cards (Blue)

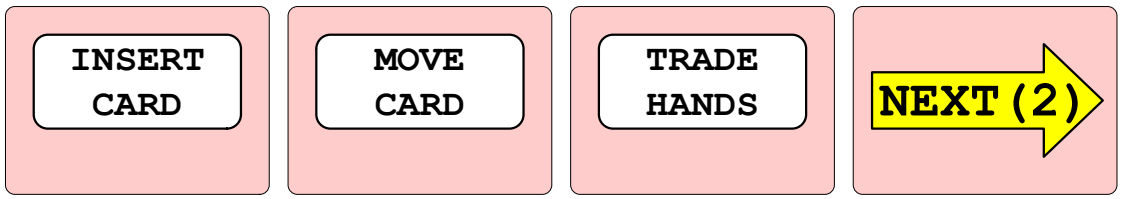
Instruction Cards have an I in the upper left corner and have a blue background. They represent instructions (statements) that make up the common program. There are two types of instruction cards: Assignments (indicated by an equals sign =) and Control Flow Instructions (indicated by a

circular arrow). Sample Instruction Cards appear below.



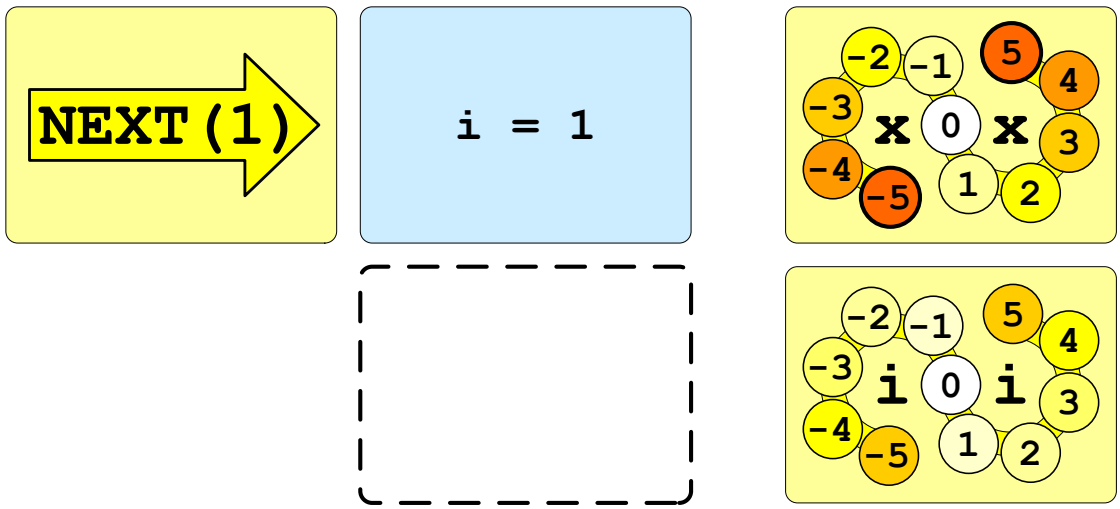
Special Action Cards (Pink)

Special Action Cards have an A in the upper left corner and have a pink background. They allow the player to perform a special action such as modifying the program. The number beneath the A indicates the number of actions required to play the card (explained below). An ✕ below the number indicates that at least one instruction does not execute normally at the end of this turn. Sample Special Action Cards appear below.



Beginning the Game

To begin the game, the Start Cards are arranged as shown:



The Top Card (the instruction "`i = 1`") is placed on the gaming table, and the NEXT Instruction Pointer card is placed to its left. As the game is played, additional Instruction Cards will be placed on the table below the Top Card, and this chain of cards will be referred

to as "the program". The two counter cards (***x*** and ***i***) are placed to the right of the Top Card, and a marker is placed at zero on each counter card to indicate the current value of that variable. The remaining green-backed cards are shuffled, and each player is randomly assigned a different sign, **Positive** or **Negative**, indicating the direction in which they are attempting to move the variable ***x***. Negative is dealt **4** cards and plays the 1st turn, 3rd turn, 5th turn, etc. Positive is dealt **5** cards and plays the 2nd turn, 4th turn, etc.

Turn Sequence

On a player's turn she **takes TWO actions**, and then **advances the NEXT pointer(s)**. The key concept of advancing a NEXT pointer is

explained later. The player may choose from five types of actions:

1. Play an Instruction

The player places an Instruction Card from her hand on the table at the bottom of the program. If the card above contains a downward blue arrow  , the new card must be indented such that its left side aligns with that arrow. Otherwise, the card may be placed at any level of indentation between that of the Top Card and that of the card above.

2. Play a Special Action

Special Action Cards allow the player to do something other than adding to the end of the program. Examples of special actions include adding or removing Instruction

Cards from the program, introducing a new NEXT pointer, and trading hands. Each special action is explained by the text on the face of the card.

INSERT, **MOVE**, and **DELETE** cards can be applied to **any Instruction Card** in the program as long as (A) **no NEXT pointer points to the card** being moved or deleted, and (B) the indentation rules described above can be satisfied simply by adjusting indentations. Following the modification, slide cards up or down so that each card sits below its predecessor (NEXT pointers travel with the card to which they originally pointed). Then make any indentation adjustments required to avoid violating indentation rules,

working from the top of the program downward. If a NEXT pointer pointed to the space below the program, then an instruction moved into the empty space will be pointed to and will execute during the "Advance NEXT pointers" phase.

SET NEXT and **NEW THREAD** cards can be used to position a NEXT pointer at **any Instruction Card**, regardless of what conditions appear above it. (That NEXT pointer does not advance or execute during the "Advance NEXT" phase of the current turn.)

Note that some powerful Special Action Cards require the player to spend both actions in order to play the card (indicated on the card by a numeral 2).

3. Draw a Card

If the player's hand contains **less than 5 cards**, he or she may draw a new card. If the deck is empty, the discard pile is reshuffled to become the new deck.

4. Discard a Card

The player may discard a card. (Drawing a replacement card is a separate action.)

5. Advance any Next Pointer

A player can opt to use an action to advance *any one* (not all) NEXT pointers. This is in addition to the advancement of pointers that must occur at the end of the turn.

Ending the Game

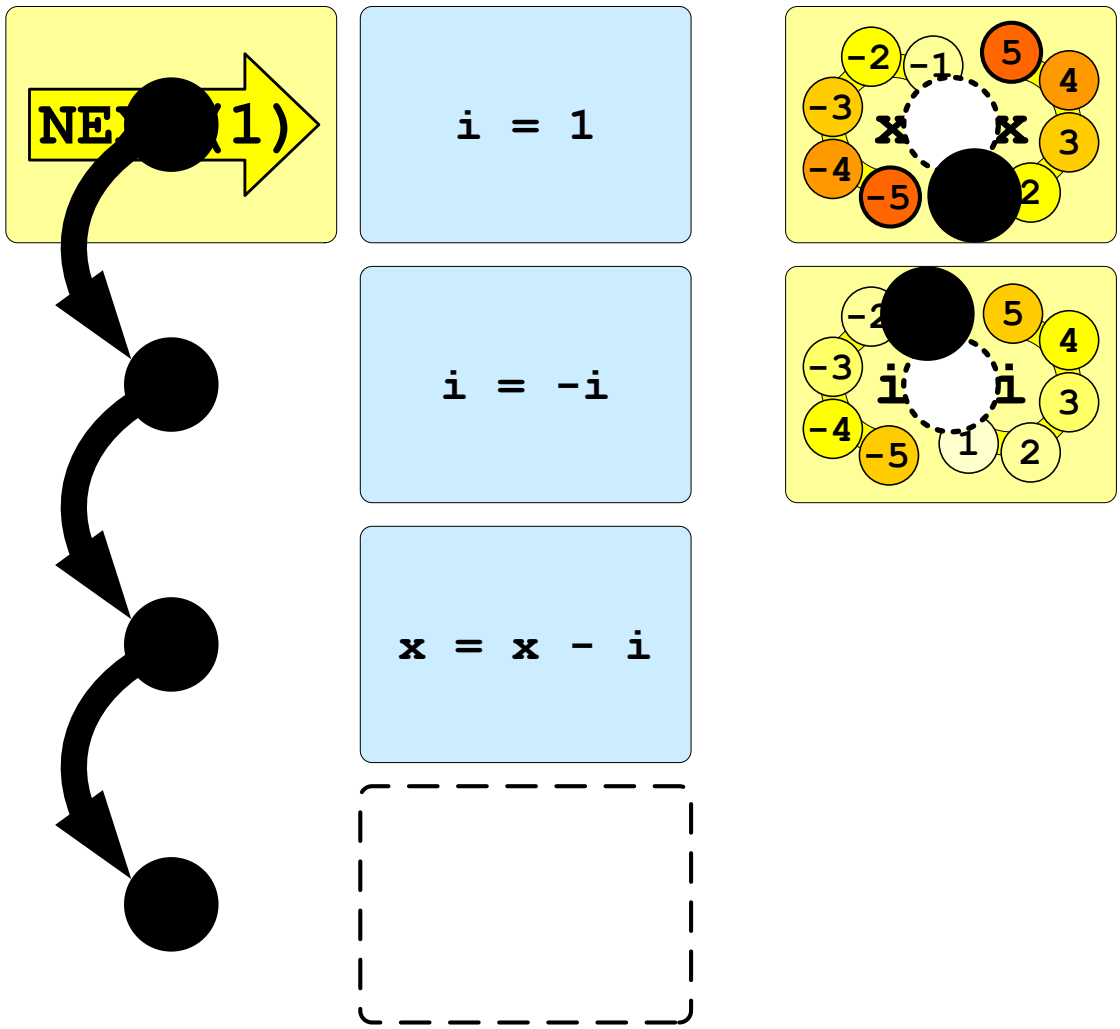
The game ends when the value of the variable ***x*** reaches or exceeds 5

(in which case Positive wins), or -5 (in which case Negative wins). This immediately ends the game.

Executing the Program (Advancing the NEXT Pointers)

The most fundamental concept in Race Condition™ is that of advancing a NEXT instruction pointer, or in other words executing the current program. Unless the NEXT pointer points to the empty space below the program, this is done by (A) executing (performing the action described on) the Instruction Card pointed to by the NEXT pointer, and (B) moving the NEXT pointer to point to the next instruction to be executed. This is often the

following instruction, but not always. A simple sequence of instructions is depicted below.



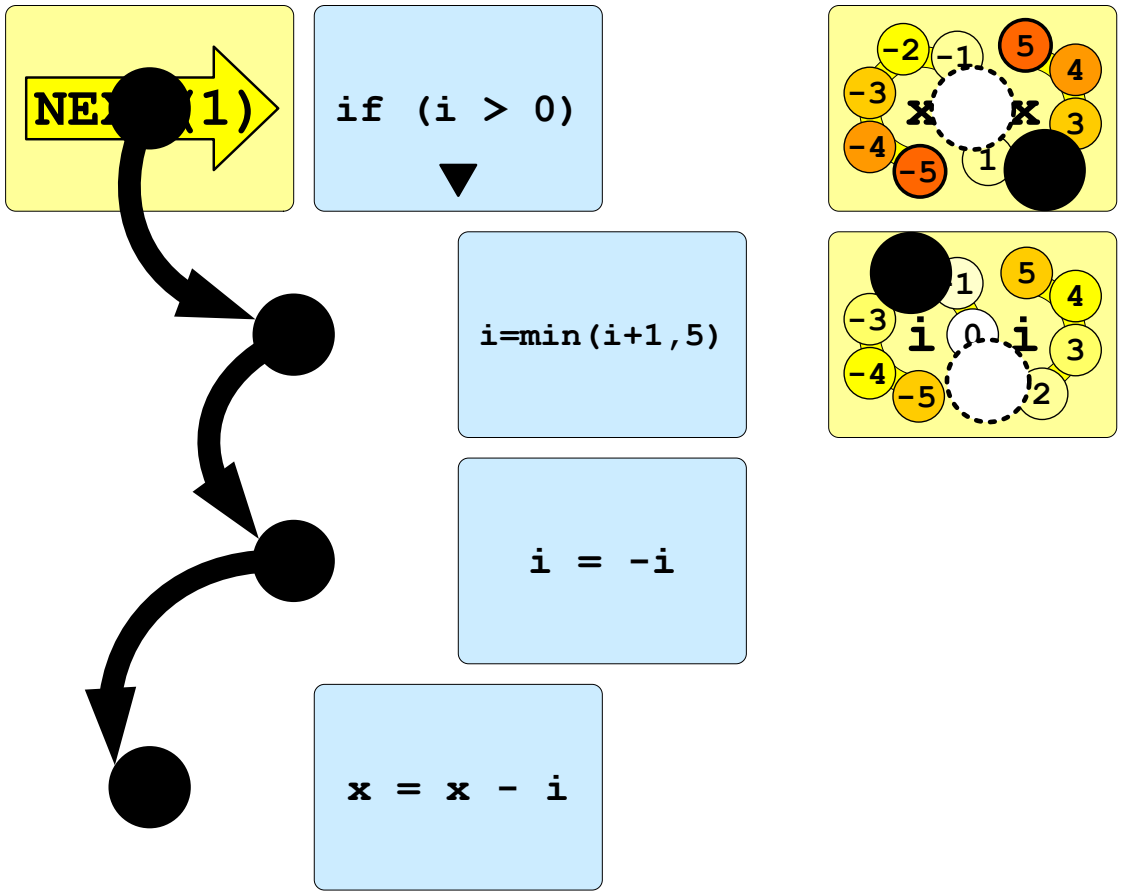
In this example, advancing the NEXT pointer consists of executing the instruction to which it points, and then sliding it down to the next card. Advancing it three times

results in executing "`i = 1`" (setting ***i*** to 1), then executing "`i = -i`" (setting ***i*** to -1), and lastly executing "`x = x - i`" (setting ***x*** to 1). This leaves the NEXT pointer pointing to the empty space just below the last card in the program. Once the NEXT pointer is pointing to this empty space, it ceases to advance until either a new instruction is played into the empty space, or a special action has the effect of moving the NEXT pointer to point to an instruction.

Conditionals

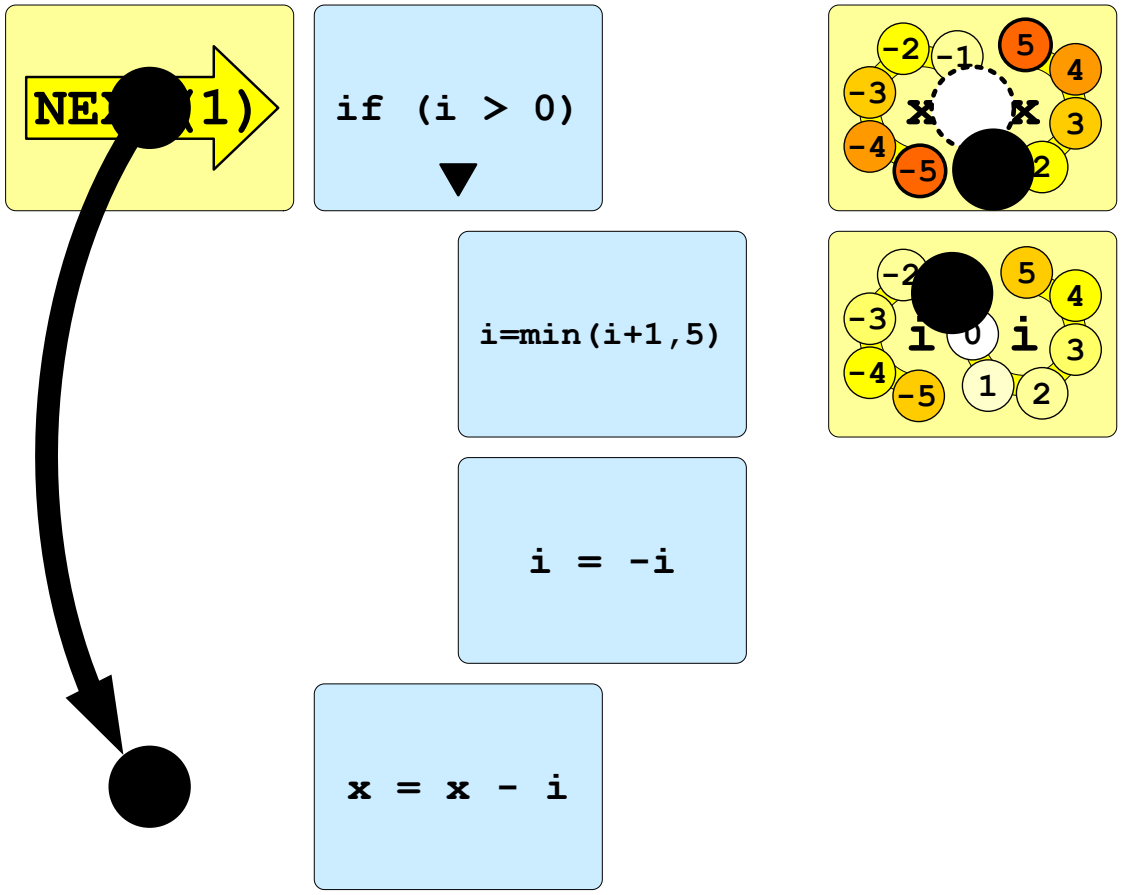
An "if" card represents a conditional: a special instruction that affects the flow of control. If the condition that appears in parentheses after the "if" is true, then when the NEXT pointer

advances, its next stop is the instruction indented under the "if" card. This procession is illustrated below.



If the condition is false, however, then the set of cards indented below the "if" (known as the "if" block) is skipped. The NEXT pointer advances as if the indented

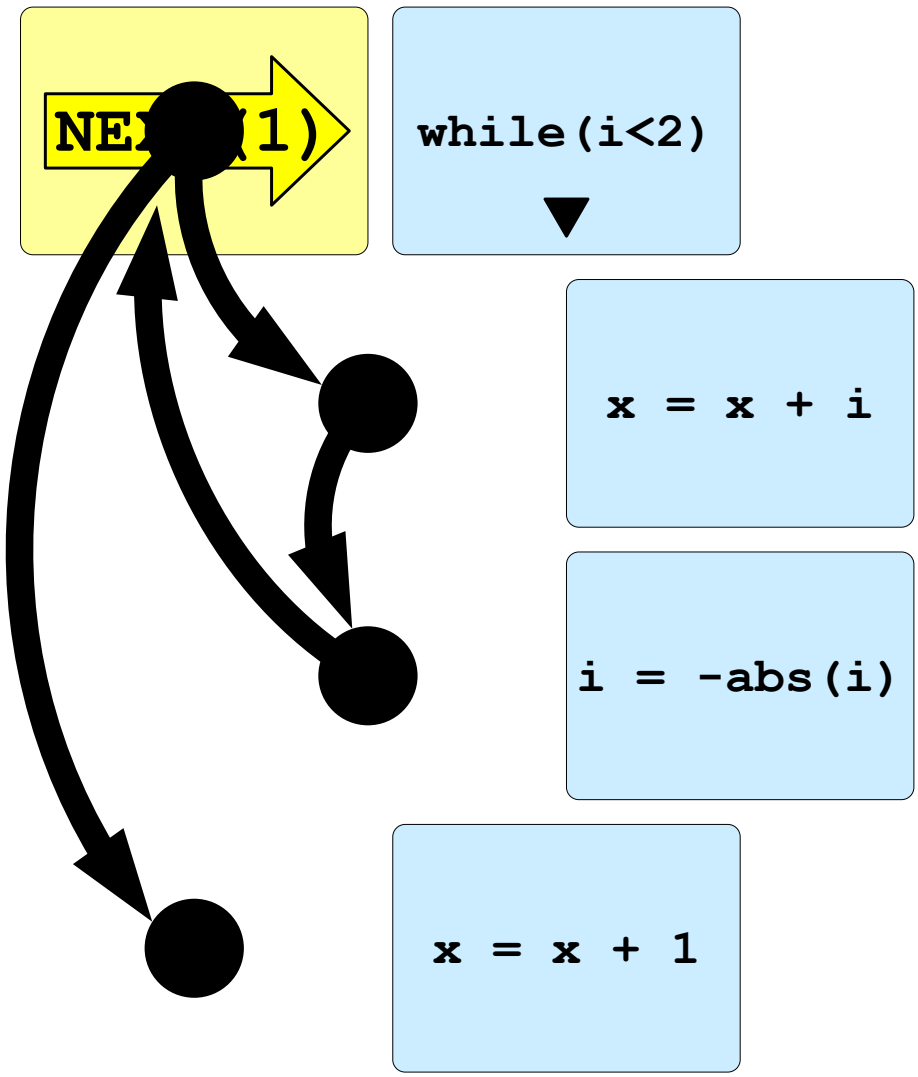
instructions did not exist, as shown.



While Loops

A "while" card represents another type of instruction that affects the flow of control. As with an "if" card, the cards indented beneath the "while" card are only visited if the condition in parentheses is true.

Unlike in an "if" block, however, at the end of a closed "while" block, the NEXT pointer returns to the "while" card at the top of the loop. This procession is illustrated below.

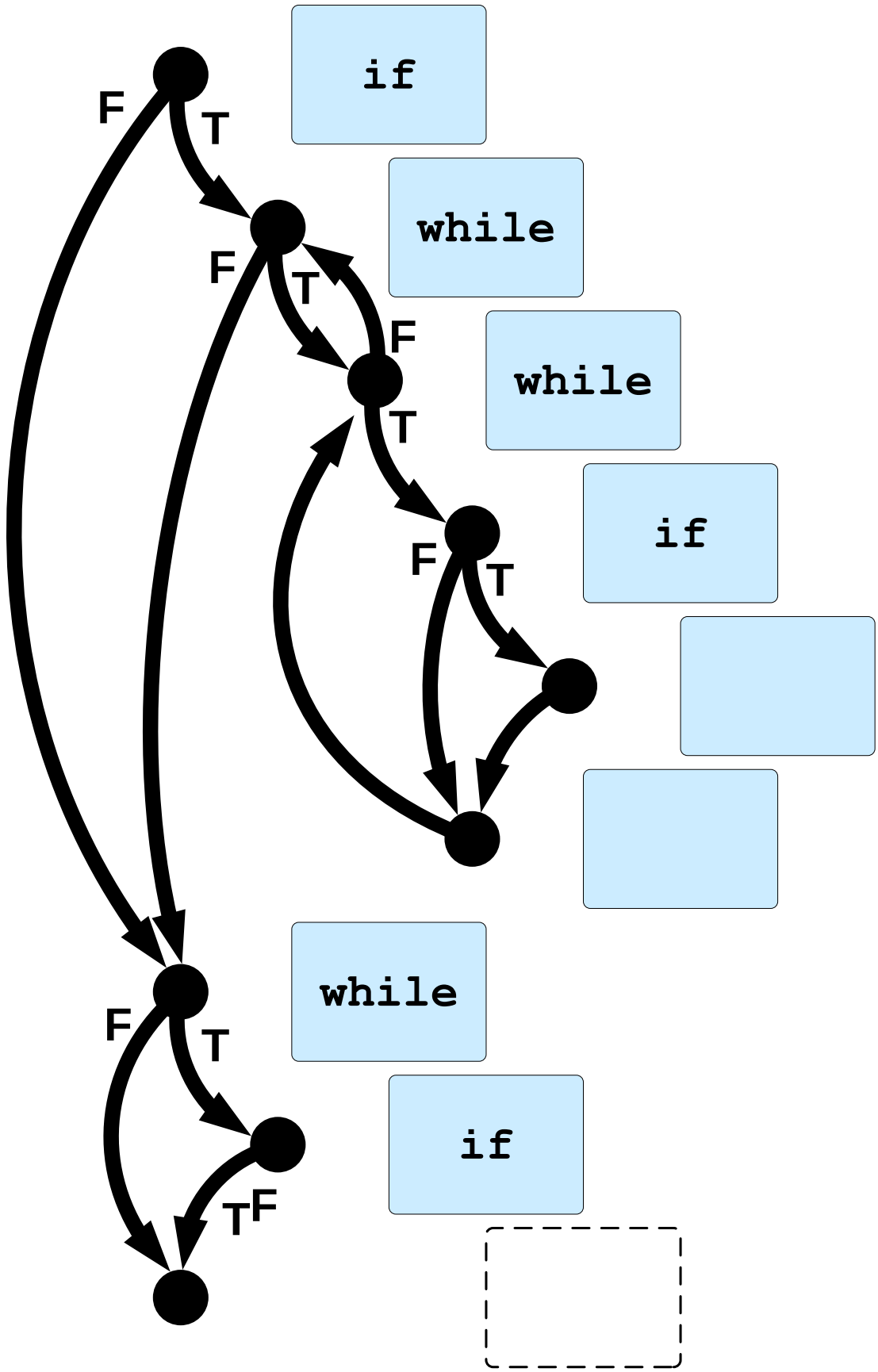


If the last card in the indented block is an Assignment (=) (as

must be the case if any cards exist later in the program at the same indentation as or to the left of the "while" card), then the "while" block is a "closed while loop," and from the last card in the block the NEXT pointer advances to the "while" card above. Otherwise, the loop is "open" and the NEXT pointer advances to the empty space below the last card.

Nested Control Flow

When an "if" or a "while" are played indented beneath another "if" or "while", they are said to be "nested". An example of nesting is shown below:


if
while
while
if
while
if
F
T
F
T
F
T
F
T
F
T
F
T

Next Instruction Summary

In summary: if the NEXT pointer points to the space below the program, nothing happens; otherwise the next instruction visited after executing any instruction is determined by the indentation of the card below it.

- **Same indentation:** Advance to the card below.
- **Greater indent (to the right):** If the relevant condition is true, advance to the instruction indented below the current instruction. Otherwise, advance as if the instructions in the indented block did not exist.
- **Lesser indent (to the left):** If moving to the card below would mean exiting a while loop, move

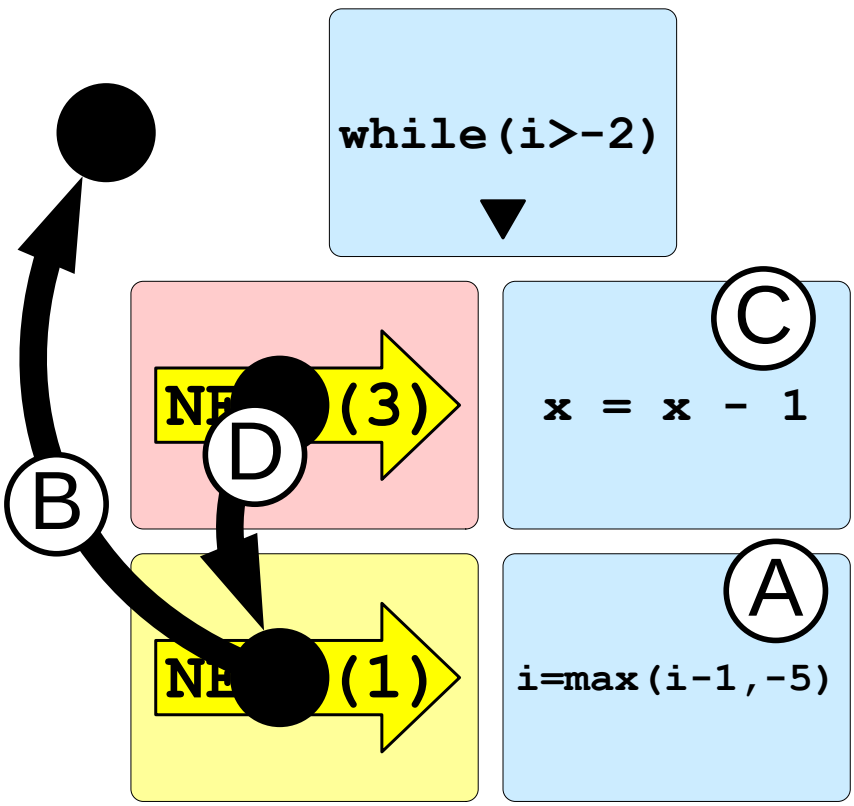
to the (rightmost) "while" card above. Otherwise, advance to the card below.

- **No cards below:** If the current instruction is an Assignment  *and* is contained within a while loop, then advance to the "while" card above. Otherwise, advance to the empty space below the last card in the program.

Multiple Threads (Multiple NEXT Pointers)

It is possible for multiple NEXT pointers to simultaneously execute the same program, and to interact with each other. These are known as "threads", and Special Action Cards can introduce new threads. If two NEXT pointers point to the same instruction, position the

cards such that each arrow's priority number is visible. In the scenario depicted below, when all threads advance, first the "**NEXT (1)**" pointer executes (A) and moves (B), and then the "**NEXT (3)**" pointer executes (C) and moves (D).



http://multinationalgames.com