// C A R D C O M P A R I S O N |

//-------------------------------------------- DO NOT CTRL A, CTRL C, CTRL V; there is C++ code at the end

MOV R3, #63 // Move '?' to register R4

MOV R4, #32 // Move Space to register R4

MOV R5, #62 // Move '>' to register R5

MOV R6, #60 // Move '<' to register R6

MOV R7, #61 // Move '=' to register R7

// Letters for comparison messages

MOV R9, #67 // Move 'C' to register R9

MOV R10, #49 // Move '1' to register R10

MOV R11, #50 // Move '2' to register R11

MOV R12, #58 // Move ':' to register R12

// Letters for output messages "C1:" and "C2:"

// Getting card values & checking for input errors

// P A R T 2 – Verify the Values

// Part 2 Report: The values are inputted, then compared to 13 and 1. If they are greater than 13 or less than 1, we output an error and end the execution. The error label is at the end of this file, so that we are sure to terminate. If the data is appropriate, we output it for visibility and go on with the execution.

OUT R9,7 // Output "C1:"

OUT R10,7

OUT R12,7

INP R0,2 // Input the 1st card

CMP R0,#13 // Compate the input card to 13, to see if it is <= than 13

BGT InputError // If not the case, call InputError

CMP R0,#1 // Compate the input card to 1, to see if it is >= than 1

BLT InputError // If not the case, call InputError

OUT R0,4 // Outout the value of the card

OUT R4, 7 // Output space

OUT R3, 7 // Output '?'

OUT R4, 7 // Output space

OUT R9,7 // Output "C2:"

OUT R11,7

OUT R12,7

INP R1,2 // Input the 2nd card

CMP R1,#13 // Compate the input card to 13, to see if it is <= than 13

BGT InputError // If not the case, call InputError

CMP R1,#1 // Compate the input card to 1, to see if it is >= than 1

BLT InputError // If not the case, call InputError

OUT R1,4 // Outout the value of the card

// Convert to Ascii

// R2 and R8 are the only 2 empty registers at this point (The rest are actually necessary, they should stay)

MOV R2, R0 // Copy numerical value of card 1

MOV R8, R1 // Copy numerical value of card 2

ADD R0, R0, #48 // Convert R0 to Ascii

ADD R1, R1, #48 // Convert R1 to Ascii

// !! NOTE: Now all the numbers will be 48 greater than their input !!

CMP R0, #49 // Check if card 1 is an Ace

BNE Continue1 // If not, continue

MOV R0, #65

Continue1:

CMP R0, #59 // Check if card 1 is a J

BNE Continue2 // If not, continue

MOV R0, #74

Continue2:

CMP R0, #60 // Check if card 1 is a Q

BNE Continue3 // If not, continue

MOV R0, #81

Continue3:

CMP R0, #61 // Check if card 1 is a K

BNE Continue4 // If not, continue

MOV R0, #75

Continue4:

// By now, the 1st card should have converted to its Ascii corespondent, or into either J,Q,K or Ace

CMP R1, #49 // Check if card 2 is an Ace

BNE Continue5 // If not, continue

MOV R1, #65

Continue5:

CMP R1, #59 // Check if card 2 is a J

BNE Continue6 // If not, continue

MOV R1, #74

Continue6:

CMP R1, #60 // Check if card 2 is a Q

BNE Continue7 // If not, continue

MOV R1, #81

Continue7:

CMP R1, #61 // Check if card 2 is a K

BNE Continue8 // If not, continue

MOV R1, #75

Continue8:

// By now, the 2nd card should have converted to its Ascii corespondent, or into either J,Q,K or Ace

// Results

// P A R T 3 – Output Card Value (character or number)

// Part 3 Report: Since we have stored both the numerical and "card value" ('A', 'J', 'Q', 'K', '10', '9', etc.) of the cards, it will be easy to compare using 2 numeric registers (to call the so), then use separate registers to output the "card value" with. The pseudocode is written in C++ and had 3 if statements, deciding which comparison statement to print.

// The isTen and Skip labels were added because number 10 cannot be stored as an Ascii character, so it has to be printed as a number.

OUT R9,7 // Output "C1:"

OUT R10,7

OUT R12,7

OUT R4, 7 // Output space

CMP R0, #58 // Vefify if card 1 is not 10

BEQ isTen1

OUT R0,7 // Output card 1, unless it is 10

B Skip1

isTen1:

OUT R2,4 // Output card 1, if it is equal to 10

Skip1:

OUT R4, 7 // Output space

// P A R T 1 – Compare Values

// Part 1 Report: By comparing the numerical values of the cards, we decide which label we should go to. By default, we continue the code as is, assuming the cards are equal (which is the only possibility left). We output the corresponding comparison sign ('>', '<' or '='), and finish off the execution by outputting the value of the last card.

CMP R2,R8 // !! NOTE: We have to compare them as numbers!

BLT Smaller // Should we go to the 'Smaller' label?

BGT Greater // Should we go to the 'Greater' label?

OUT R7, 7 // Well, if not, print '=' and continue this set of instructions

OUT R4, 7 // Output space

OUT R9,7 // Output "C2:"

OUT R11,7

OUT R12,7

OUT R4, 7 // Output space

CMP R1, #58 // Vefify if card 1 is not 10

BEQ isTen2

OUT R1,7 // Output card 2, unless it is 10

B Skip2

isTen2:

OUT R8, 4 // Output card 2, if it is equal to 10

Skip2:

B DONE

DONE:

HALT

Smaller:

OUT R6, 7 // Output '<'

OUT R4, 7 // Output space

OUT R9,7 // Output "C2:"

OUT R11,7

OUT R12,7

OUT R4, 7 // Output space

CMP R1, #58 // Vefify if card 1 is not 10

BEQ isTen3

OUT R1,7 // Output card 2, unless it is 10

B Skip3

isTen3:

OUT R8, 4 // Output card 2, if it is equal to 10

Skip3:

B DONE

Greater:

OUT R5, 7 // Output '>'

OUT R4, 7 // Output space

OUT R9,7 // Output "C2:"

OUT R11,7

OUT R12,7

OUT R4, 7 // Output space

CMP R1, #58 // Vefify if card 1 is not 10

BEQ isTen4

OUT R1,7 // Output card 2, unless it is 10

B Skip4

isTen4:

OUT R8, 4 // Output card 2, if it is equal to 10

Skip4:

B DONE

InputError:

MOV R9, #69 // Move 'E' to register R9

MOV R10, #114 // Move 'r' to register R10

MOV R11, #111 // Move 'o' to register R11

// Letters for error code

OUT R9,7 // Output "Error"

OUT R10,7

OUT R10,7

OUT R11,7

OUT R10,7

// C A R D C O M P A R I S O N |

//--------------------------------------------

// C P P P S E U D O C O D E //

#include <iostream>

using namespace std;

int a, b; // The card numbers

char aChar, bChar; // The Ascii character correspondants

void ComputeCard(int &x, char &xChar) // This accepts parameters for both card 1 and 2 (note the &)

{

cout<<"Input card: ";

cin>>x;

if ( ! ( x>=1 && x<=13 ) ) // Verification of input

{

cout<<"Please input a number between 1 and 13 inclusivly.";

std::exit(0); // As specified during the course, the execution should end onInvalidInput

}

else

{

if ( x>=2 && x<=10 )

xChar = 48 + x;

Else

{

if (x==11)

xChar='J';

if (x==12)

xChar='Q';

if (x==13)

xChar='K';

if (x==1)

xChar='A';

}

}

}

// P A R T 3 P S E U D O C O D E

void Output() // Compare a and b (the numerical values), to output the “card values”

{

if (a == b)

cout<<"Card "<<aChar<<" is IDENTICAL to card "<<bChar;

if (a < b)

cout<<"Card "<<aChar<<" is SMALLER than card "<<bChar;

if (a > b)

cout<<"Card "<<aChar<<" is LARGER than card "<<bChar;

}

int main()

{

ComputeCard(a, aChar); // Compute card 1

ComputeCard(b, bChar); // Compute card 2

Output();

return 0;

}