; CIS 11 FINAL PROJECT

; OPTION B: TEST SCORE CALCULATOR

; TEAM 2: MUSAMIM MUBTAKIR

; JOSHUA SEPULVEDA

; START PROGRAM

.ORIG x3000

; MAIN

LEA R0, PROMPT

PUTS

PROMPT .STRINGZ "Enter 5 test scores [0 – 99]: "

LD R0, NEWLINE

OUT

; JUMPS TO GET\_GRADE TO GET NUMERIC GRADE, STORE

; INTO GRADES ARRAY. JUMP TO GET LETTER FUNCTION AND

; THEN POP FUNCTION. REPEATED FOR ALL TEST SCORES.

JSR GET\_GRADE

LEA R6, GRADES

STR R3, R6, #0

JSR GET\_LETTER

JSR POP

LD R0, NEWLINE

OUT

JSR GET\_GRADE

LEA R6, GRADES

STR R3, R6, #1

JSR GET\_LETTER

JSR POP

LD R0, NEWLINE

OUT

JSR GET\_GRADE

LEA R6, GRADES

STR R3, R6, #2

JSR GET\_LETTER

JSR POP

LD R0, NEWLINE

OUT

JSR GET\_GRADE

LEA R6, GRADES

STR R3, R6, #3

JSR GET\_LETTER

JSR POP

LD R0, NEWLINE

OUT

JSR GET\_GRADE

LEA R6, GRADES

STR R3, R6, #4

JSR GET\_LETTER

JSR POP

LD R0, NEWLINE

OUT

; CALCULATE MAX

CALCULATE\_MAX

LD R1, NUM\_TESTS ; R1 HOLDS THE TOTAL NUMBER OF TEST

LEA R2, GRADES ; R2 HOLDS THE STARTING ADDRESS OF GRADES

LD R4, GRADES

ST R4, MAX\_GRADE

ADD R2, R2, #1

LOOP1 LDR R5, R2, #0 ; ACCESS POINTER VALUE IN GRADES

NOT R4, R4

ADD R4, R4, #1

ADD R5, R5, R4

BRp NEXT1

LEA R0, MAX

PUTS

LD R3, MAX\_GRADE

AND R1, R1, #0

JSR BREAK\_INT

LD R0, SPACE

OUT

LD R0, NEWLINE

OUT

JSR CLEAR\_REG

; CALCULATE MIN

CALCULATE\_MIN

LD R1, NUM\_TESTS ; R1 HOLDS THE TOTAL NUMBER OF TEST

LEA R2, GRADES ; R2 HOLDS THE STARTING ADDRESS OF GRADES

LD R4, GRADES

ST R4, MIN\_GRADE

ADD R2, R2, #1

ADD R1, R1, #-1

LOOP2 LDR R5, R2, #0 ; ACCESS POINTER VALUE IN GRADES

NOT R4, R4

ADD R4, R4, #1

ADD R5, R5, R4

BRn NEXT2

ADD R2, R2, #1

LD R4, GRADES

AND R5, R5,#0

ADD R1,R1,#-1

BRp LOOP2

LEA R0, MIN

PUTS

LD R3, MIN\_GRADE

AND R1, R1, #0

JSR BREAK\_INT

LD R0, SPACE

OUT

JSR CLEAR\_REG

LD R0, NEWLINE

OUT

; CALCULATE AVG

CALC\_AVG

LD R1, NUM\_TESTS ; R1 HOLDS THE TOTAL NUMBER OF TEST

LEA R2, GRADES ; R2 HOLDS THE STARTING ADDRESS OF GRADES

GEN\_SUM LDR R4, R2, #0

ADD R3, R3, R4 ; R3 STORES THE SUM

ADD R2, R2, #1

ADD R1, R1, #-1

BRp GEN\_SUM

LD R1, NUM\_TESTS

NOT R1, R1

ADD R1, R1, #1 ; R1 = #-5

ADD R4, R3, #0

LOOP3 ADD R4, R4, #0

BRnz DONE\_AVG

ADD R6, R6, #1 ; INCR

ADD R4, R4, R1 ; SUBRACT 5 FROM TOTAL

BRp LOOP3

DONE\_AVE

ST R6, AVERAGE\_SCORE

LEA R0, AVG

PUTS

AND R3, R3, #0

AND R1, R1, #0

AND R4, R4, #0

ADD R3, R3, R6

JSR BREAK\_INT

JSR RESTART\_PROG

HALT

; END MAIN

NEWLINE .FILL xA

SPACE .FILL X20

DECODE\_DEC .FILL #-48

DECODE\_SYM .FILL #48

DECODE\_THIRTY .FILL #-30

NUM\_TESTS .FILL #5

RESTART2 .FILL x3000

MAX\_GRADE .BLKW x1

MIN\_GRADE .BLKW x1

DONE\_AVG .BLKW x1

AVERAGE\_SCORE .BLKW x1

; BRANCHES AND VARIABLES TO CALCULATING MIN & MAX

NEXT2

LDR R4, R2, #0

ST R4, MIN\_GRADE

ADD R2, R2, #1 ; GRADES ARRAY MOVE UP

ADD R1, R1, #-1 ; COUNTER MOVE DOWN

BRnzp LOOP2

NEXT1

LDR R4, R2, #0

ST R4, MAX\_GRADE

ADD R2, R2, #1 ; GRADES ARRAY MOVE UP

ADD R1, R1, #-1 ; COUNTER MOVE DOWN

BRnzp LOOP1

GRADES .BLKW #5

MIN .STRINGZ "MIN "

MAX .STRINGZ "MAX "

AVG .STRINGZ "AVG "

; SUBROUTINE: RESTART PROGRAM

RESTART\_PROG

ST R7, SAVELOC1 ; SAVE JSR LOCATION

LD R1, LOWER\_Y ; LOAD NEG VALUE OF Y

LD R3, UPPER\_Y

LD R2, ORIGIN ; LOAD ORIGIN ( x3000)

LD R0, NEWLINE

OUT

LEA R0 RESTARTPROG\_STR ; RESTART PROMPT STRING

PUTS

LD R0, NEWLINE

OUT

GETC

ADD R1, R1, R0 ; COMPARE USER INPUT WITH -y

BRz RESTART\_TRUE ; IF TRUE BRANCH TO RESTART

ADD R3, R3, R0 ; COMPARE USER INPUT WITH -Y

BRz RESTART\_TRUE ; IF TRUE BRANCH TO RESTART

HALT ; ELSE HALT PROGRAM

; RESTART\_PROG BRANCHES

RESTART\_TRUE

JMP R2

RESTARTPROG\_STR .STRINGZ "PROGRAM HAS ENDED, WOULD YOU LIKE TO TEST MORE SCORES? Y/N "

LOWER\_Y .FILL xFF87 ; -121

UPPER\_Y .FILL xFFA7 ; -89

ORIGIN .FILL x3000

; END RESTART PROGRAM

SAVELOC1 .FILL X0

SAVELOC2 .FILL X0

SAVELOC3 .FILL X0

SAVELOC4 .FILL X0

SAVELOC5 .FILL X0

; SUBROUTINE: GET\_GRADE

GET\_GRADE ST R7, SAVELOC1 ; STORE JSR LOCATION

JSR CLEAR\_REG ; CLEAR REGISTERS

LD R4, DECODE\_DEC ; LOAD CONVERSION

GETC ; GET FIRST CHAR

JSR VAL

OUT ; ECHO INPUT

ADD R1, R0, #0 ; COPY INPUT TO R1

ADD R1, R1, R4 ; CONVERT TO DECIMAL

ADD R2, R2, #10 ; CLEAR R2

MULT10 ADD R3, R3, R1 ; ADD INPUT TO R3 (MULT PROCESS)

ADD R2, R2, #-1 ; DECR COUNTER

BRp MULT10 ; LOOP UNTIL COUNTER IS ZERO

GETC ; GET SECOND CHAR

JSR VAL

OUT ; ECHO INPUT TO SCREEN

ADD R0, R0, R4 ; CONVERT SECOND INPUT TO DECIMAL

ADD R3, R3, R0 ; ADD FIRST INPUT TO SECOND INPUT

LD R0, SPACE ; ADD SPACE

OUT ; PRINT SPACE

LD R7, SAVELOC1 ; LOAD JSR RETURN LOCATION

RET ; RETURN

; END GET\_GRADE

; SUBROUTINE: BREAK \_ INT

BREAK\_INT

ST R7, SAVELOC1 ; STORE JSR RETURN LOCATION

LD R5, DECODE\_SYM ; CONVERT DECIMAL TO SYMBOL

ADD R4, R3, #0 ; COPY INPUT TO R4 (PLATFORM)

DIV1 ADD R1, R1, #1 ; COUNTER FOR DIVISION

ADD R4, R4, #-10 ; SUBTRACT 10 FROM INPUT

BRp DIV1 ; SUBTRACT 10 TILL INPUT IS 0 OR NEG

ADD R1, R1 #-1 ; REMOVE EXTRA 1

ADD R4, R4, #10 ; ADD 10 TO GET REMAINDER

ADD R6, R4, #-10

BRnp POS

NEG ADD R1, R1, #1

ADD R4, R4, #-10

POS ST R1, Q ; STORE QUOTIENT

ST R4, R ; STORE REMAINDER (MOD 10)

LD R0, Q ; LOAD QUOTIENT FOR PRINT

ADD R0, R0, R5 ; CONVERT DECIMAL TO SYMBOL

OUT ; PRINT QUOTIENT

LD R0, R ; LOAD REMAINDER FOR PRINT

ADD R0, R0, R5 ; CONVERT DECIMAL TO SYMBOL

OUT ; PRINT REMAINDER

LD R7, SAVELOC1 ; RESTORE JSR RETURN LOCATION

RET

R .FILL X0

Q .FILL X0

; END BREAK\_INT

; SUBROUTINE: PUSH

PUSH ST R7, SAVELOC2 ; STORE JSR LOCATION

JSR CLEAR\_REG ; CLEAR REGISTERS

LD R6, POINTER ; INITIALIZE POINTER

ADD R6, R6, #0

BRnz STACK\_ERROR

ADD R6, R6, #-1 ; DECR POINTER

STR R0, R6, #0 ; STORE NUMBER IN R0 TO STACK

ST R6, POINTER ; SAVE POINTER LOCATION

LD R7, SAVELOC2 ; RESTORE LOCATION

RET

POINTER .FILL X4000 ; POINTER START LOCATION

; END PUSH

; SUBROUTINE: POP

POP LD R6, POINTER ; LOAD POINTER LOCATION

ST R1, SAVELOC5

LD R1, BASELINE

ADD R1, R1, R6

BRzp STACK\_ERROR

LD R1, SAVELOC5

LDR R0, R6, #0 ; LOAD VALUE IN STACK INTO R0

ST R7, SAVELOC4 ; STORE JSR LOCATION

OUT ; PRINT NUMBER FROM STACK

LD R0, SPACE ; LOAD A SPACE

OUT ; PRINT SPACE

ADD R6, R6, #1 ; INCR POINTER

ST R6, POINTER ; STORE POINTER LOCATION

LD R7, SAVELOC4

RET

STACK\_ERROR LEA R0, ERROR

PUTS

HALT

BASELINE .FILL xC000

ERROR .STRINGZ "STACK UNDERFLOW OR OVERFLOW!!! PROGRAM HALTING... "

; END POP

; SUBROUTINE: GET\_LETTER

GET\_LETTER

AND R2, R2, #0 ; CLEAR R2

A\_GRADE LD R0, A\_NUM ; LOAD NUMBER VALUE

LD R1, A\_LET ; LOAD SYMBOL VALUE

ADD R2, R3, R0 ; COMPARE INPUT TO VALUE OF GRADE

BRzp STR\_GRADE ; IF POS OR ZERO STORE GRADE

B\_GRADE AND R2, R2, #0

LD R0, B\_NUM

LD R1, B\_LET

ADD R2, R3, R0

BRzp STR\_GRADE

C\_GRADE AND R2, R2, #0

LD R0, C\_NUM

LD R1, C\_LET

ADD R2, R3, R0

BRzp STR\_GRADE

D\_GRADE AND R2, R2, #0

LD R0, D\_NUM

LD R1, D\_LET

ADD R2, R3, R0

BRzp STR\_GRADE

F\_GRADE AND R2, R2, #0

LD R0, F\_NUM

LD R1, F\_LET

ADD R2, R3, R0

BRNZP STR\_GRADE

RET

STR\_GRADE ST R7, SAVELOC1 ; SAVE JSR LOCATION

AND R0, R0, #0 ; CLEAR R0

ADD R0, R1, #0 ; ADD LETTER TO R0

JSR PUSH ; PUSH LETTER TO STACK

LD R7, SAVELOC1 ; RESTORE JSR LOCATION

RET ; RETURN TO MAIN

A\_NUM .FILL #-90

A\_LET .FILL X41

B\_NUM .FILL #-80

B\_LET .FILL X42

C\_NUM .FILL #-70

C\_LET .FILL X43

D\_NUM .FILL #-60

D\_LET .FILL X44

F\_NUM .FILL #-50

F\_LET .FILL X46

; END GET\_LETTER

; SUBROUTINE: CLEAR\_REG

CLEAR\_REG AND R1, R1, #0

AND R2, R2, #0

AND R3, R3, #0

AND R4, R4, #0

AND R5, R5, #0

AND R6, R6, #0

RET

; END CLEAR\_REG

; SUBROUTINE: VAL

VAL ST R1, SAVELOC5 ; STORE VARIABLES

ST R2, SAVELOC4

ST R3, SAVELOC3

LD R1, DATA\_MIN ; COMPARE INPUT TO LOWEST ACCEPTABLE DEC VALUE

ADD R2, R0, R1

BRN FAIL ; FAIL IF OUT OF RANGE

LD R1, DATA\_MAX ; COMPARE INPUT TO HIGHEST ACCEPTABLE DEC VALUE

ADD R3, R0, R1

BRP FAIL ; FAIL IF OUT OF RANGE

LD R1, SAVELOC5 ; RESTORE VARIABLES

LD R2, SAVELOC4

LD R3, SAVELOC3

RET

FAIL LEA R0, FAIL\_STR ; FAIL BRANCH

PUTS

LD R0, NEWLINE2

OUT

LD R7, RESTART ; LOAD X3000 LOCATION

JMP R7 ; RESTART PROGRAM

FAIL\_STR .STRINGZ "INVALID ENTRY!!! PROGRAM RESTARTING... "

RESTART .FILL X3000

DATA\_MIN .FILL #-48

DATA\_MAX .FILL #-57

NEWLINE2 .FILL XA

; END VAL

.END

; END PROGRAM