hroac2HW4

by Hannah Roach

Submission date: 18-Oct-2018 01:22PM (UTC-0500)

Submission ID: 1022468296

File name: hroac2HW4.pdf (1.55M)

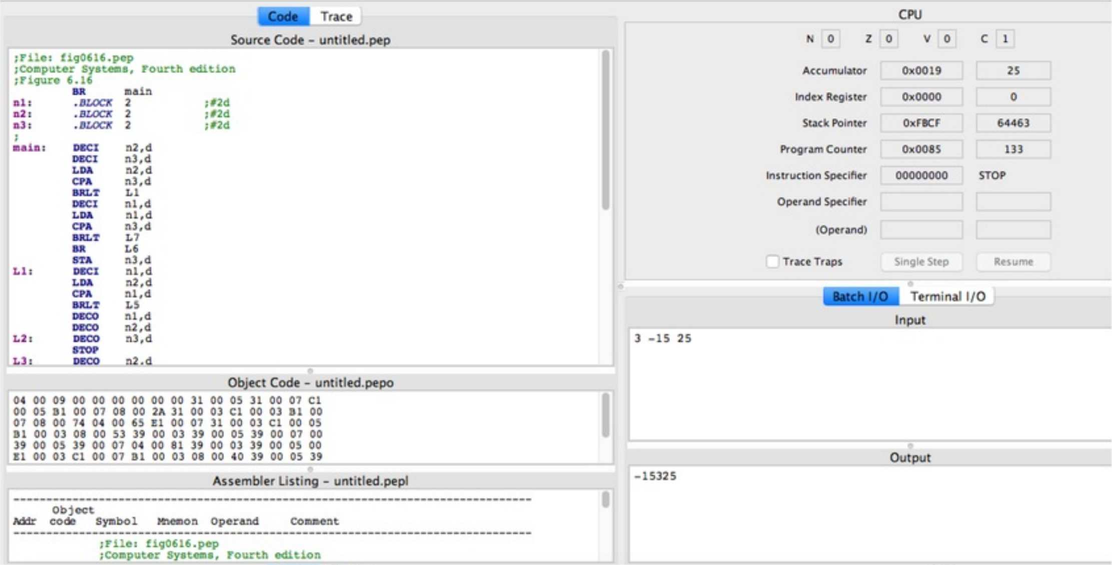
Word count: 1175

Character count: 5313

Hannah Roach
CSC 376 Computer Organization
Homework #4

Problem 1: (8 points)
Run the mystery program from Fig 6.16 with the values supplied in the help solution of Pep8 and some of your own

a) Show 3 screen shots with different inputs, including the output using Batch I/O


Code
Trace
Source Code – untitled.pep
Object Code – untitled.pepo
Assembler Listing – untitled.pepl
CPU
Batch I/O
Terminal I/O
Input
Output
Trace Traps
Single Step
Resume

```
;File: fig0616.pep
;Computer Systems, Fourth edition
;Figure 6.16
         BR      main
n1:      .BLOCK  2            ;#2d
n2:      .BLOCK  2            ;#2d
n3:      .BLOCK  2            ;#2d
;
main:    DECI    n2,d
         DECI    n3,d
         LDA     n2,d
         CPA     n3,d
         BRLT    L1
         DECI    n1,d
         LDA     n1,d
         CPA     n3,d
         BRLT    L7
         BR      L6
         STA     n3,d
L1:      DECI    n1,d
         LDA     n2,d
         CPA     n1,d
         BRLT    L5
         DECO    n1,d
         DECO    n2,d
L2:      DECO    n3,d
         STOP
L3:      DECO    n2.d
```

```
04 00 09 00 00 00 00 00 00 31 00 05 31 00 07 C1
00 05 B1 00 07 08 00 2A 31 00 03 C1 00 03 B1 00
07 08 00 74 04 00 65 E1 00 07 31 00 03 C1 00 05
B1 00 03 08 00 53 39 00 03 39 00 05 39 00 07 00
39 00 05 39 00 07 04 00 81 39 00 03 39 00 05 00
E1 00 03 C1 00 07 B1 00 03 08 00 40 39 00 05 39
```

```
-------------------------------------------------------------------------------
      Object
Addr  code   Symbol   Mnemon  Operand     Comment
-------------------------------------------------------------------------------
             ;File: fig0616.pep
             ;Computer Systems, Fourth edition
```

N 0 Z 0 V 0 C 1

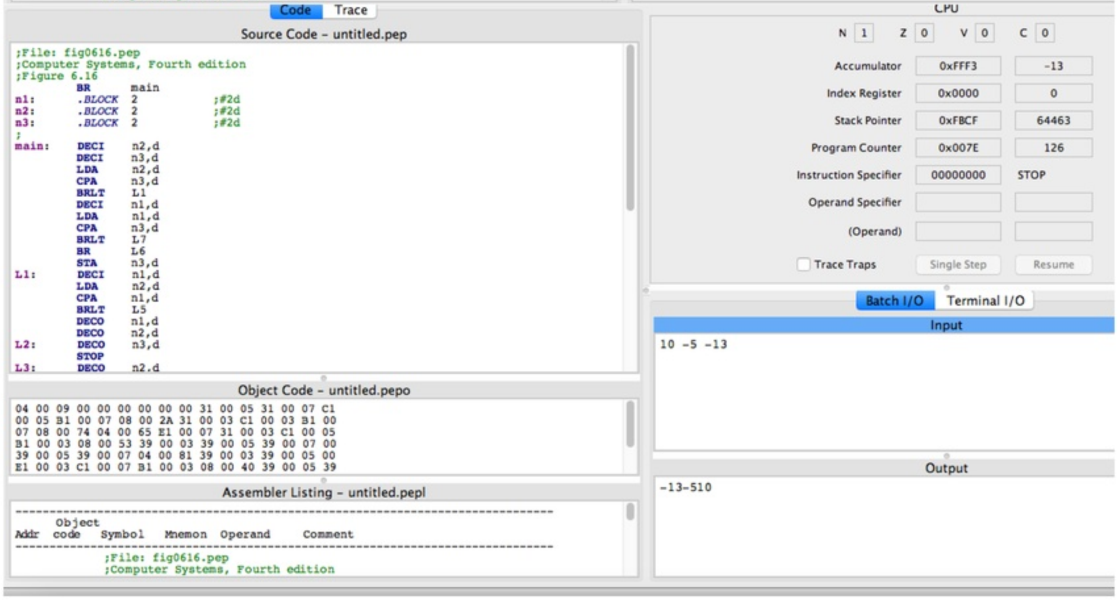
Accumulator	0x0019	25
Index Register	0x0000	0
Stack Pointer	0xFBCF	64463
Program Counter	0x0085	133
Instruction Specifier	00000000	STOP
Operand Specifier		
(Operand)		

Input

```
3 -15 25
```

Output

```
-15325
```


Code
Trace
Source Code – untitled.pep
Object Code – untitled.pepo
Assembler Listing – untitled.pepl
CPU
Batch I/O
Terminal I/O
Input
Output
Trace Traps
Single Step
Resume

```
;File: fig0616.pep
;Computer Systems, Fourth edition
;Figure 6.16
         BR      main
n1:      .BLOCK  2            ;#2d
n2:      .BLOCK  2            ;#2d
n3:      .BLOCK  2            ;#2d
;
main:    DECI    n2,d
         DECI    n3,d
         LDA     n2,d
         CPA     n3,d
         BRLT    L1
         DECI    n1,d
         LDA     n1,d
         CPA     n3,d
         BRLT    L7
         BR      L6
         STA     n3,d
L1:      DECI    n1,d
         LDA     n2,d
         CPA     n1,d
         BRLT    L5
         DECO    n1,d
         DECO    n2,d
L2:      DECO    n3,d
         STOP
L3:      DECO    n2.d
```

```
04 00 09 00 00 00 00 00 00 31 00 05 31 00 07 C1
00 05 B1 00 07 08 00 2A 31 00 03 C1 00 03 B1 00
07 08 00 74 04 00 65 E1 00 07 31 00 03 C1 00 05
B1 00 03 08 00 53 39 00 03 39 00 05 39 00 07 00
39 00 05 39 00 07 04 00 81 39 00 03 39 00 05 00
E1 00 03 C1 00 07 B1 00 03 08 00 40 39 00 05 39
```

```
-------------------------------------------------------------------------------
      Object
Addr  code   Symbol   Mnemon  Operand     Comment
-------------------------------------------------------------------------------
             ;File: fig0616.pep
             ;Computer Systems, Fourth edition
```

N 1 Z 0 V 0 C 0

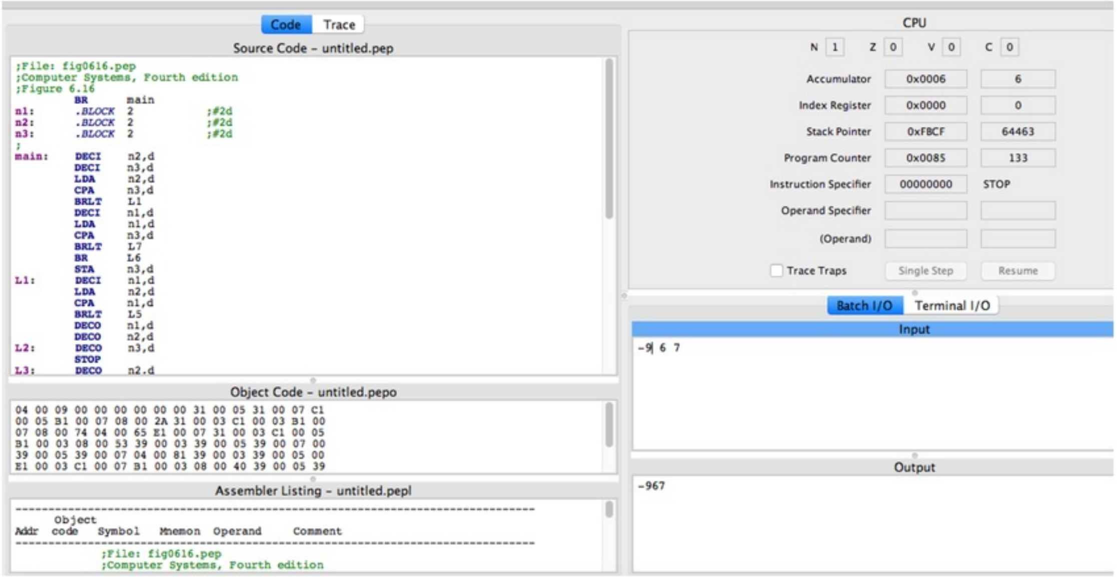
Accumulator	0xFFF3	-13
Index Register	0x0000	0
Stack Pointer	0xFBCF	64463
Program Counter	0x007E	126
Instruction Specifier	00000000	STOP
Operand Specifier		
(Operand)		

Input

```
10 -5 -13
```

Output

```
-13-510
```



Source Code – untitled.pep

```
;File: fig0616.pep
;Computer Systems, Fourth edition
;Figure 6.16
         BR      main
n1:      .BLOCK  2            ;#2d
n2:      .BLOCK  2            ;#2d
n3:      .BLOCK  2            ;#2d
;
main:    DECI    n2,d
         DECI    n3,d
         LDA     n2,d
         CPA     n3,d
         BRLT    L1
         DECI    n1,d
         LDA     n1,d
         CPA     n3,d
         BRLT    L7
         BR      L6
         STA     n3,d
L1:      DECI    n1,d
         LDA     n2,d
         CPA     n1,d
         BRLT    L5
         DECO    n1,d
         DECO    n2,d
L2:      DECO    n3,d
         STOP
L3:      DECO    n2.d
```

Object Code – untitled.pepo

```
04 00 09 00 00 00 00 00 00 31 00 05 31 00 07 C1
00 05 B1 00 07 08 00 2A 31 00 03 C1 00 03 B1 00
07 08 00 74 04 00 65 E1 00 07 31 00 03 C1 00 05
B1 00 03 08 00 53 39 00 03 39 00 05 39 00 07 00
39 00 05 39 00 07 04 00 81 39 00 03 39 00 05 00
E1 00 03 C1 00 07 B1 00 03 08 00 40 39 00 05 39
```

Assembler Listing – untitled.pepl

```
-----------------------------------------------------------------------------
      Object
Addr  code    Symbol   Mnemon  Operand     Comment
-----------------------------------------------------------------------------
               ;File: fig0616.pep
               ;Computer Systems, Fourth edition
```

N 1 Z 0 V 0 C 0

Accumulator 0x0006 6

Index Register 0x0000 0

Stack Pointer 0xFBCF 64463

Program Counter 0x0085 133

Instruction Specifier 00000000 STOP

Operand Specifier

(Operand)

☐ Trace Traps

Input

```
-9 6 7
```

Output

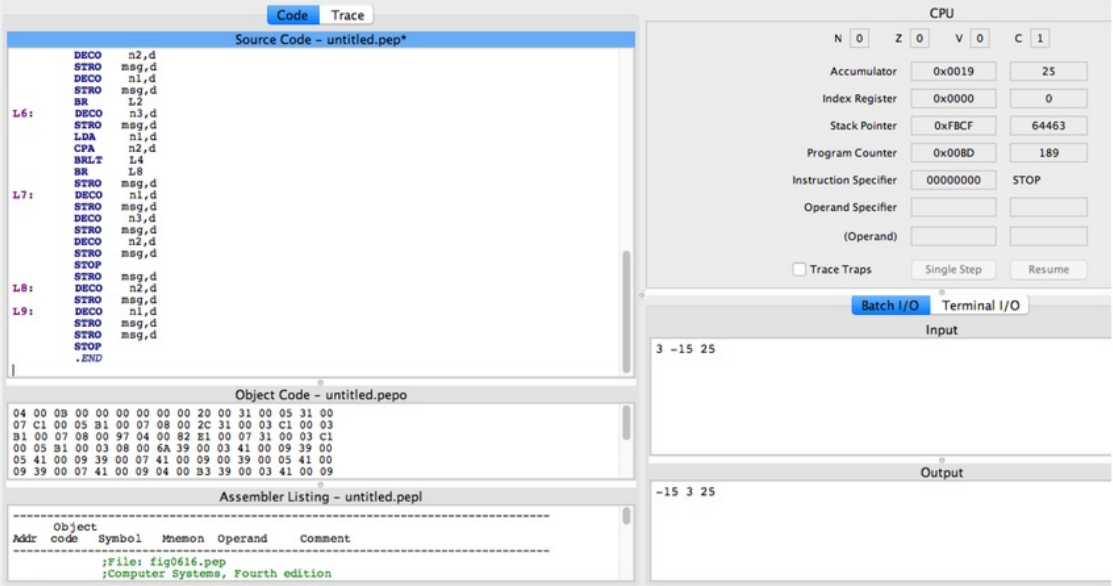
```
-967
```

b) State in one sentence, what does this program do?

The program sorts the numbers in ascending order and concatenates results.

c) Modify the program to make the output clearer – Describe – Do not paste the code
Show the same 3 screen shots with the modification

I added STRO to after DECO. I believe the same results could have been achieved with CHARO.



Source Code – untitled.pep*

```
         DECO     n2,d
         STRO    msg,d
         DECO     n1,d
         STRO    msg,d
         BR       L2
L6:      DECO     n3,d
         STRO    msg,d
         LDA      n1,d
         CPA      n2,d
         BRLT     L4
         BR       L8
         STRO    msg,d
L7:      DECO     n1,d
         STRO    msg,d
         DECO     n3,d
         STRO    msg,d
         DECO     n2,d
         STRO    msg,d
         STOP
         STRO    msg,d
L8:      DECO     n2,d
         STRO    msg,d
L9:      DECO     n1,d
         STRO    msg,d
         STRO    msg,d
         STOP
         .END
```

Object Code – untitled.pepo

```
04 00 0B 00 00 00 00 00 00 20 00 31 00 05 31 00
07 C1 00 05 B1 00 07 08 00 2C 31 00 03 C1 00 03
B1 00 07 08 00 97 04 00 82 E1 00 07 31 00 03 C1
00 05 B1 00 03 08 00 6A 39 00 03 41 00 09 39 00
05 41 00 09 39 00 07 41 00 09 00 39 00 05 41 00
09 39 00 07 41 00 09 04 00 B3 39 00 03 41 00 09
```

Assembler Listing – untitled.pepl

```
-----------------------------------------------------------------------------
      Object
Addr  code    Symbol   Mnemon  Operand     Comment
-----------------------------------------------------------------------------
               ;File: fig0616.pep
               ;Computer Systems, Fourth edition
```

N 0 Z 0 V 0 C 1

Accumulator 0x0019 25

Index Register 0x0000 0

Stack Pointer 0xFBCF 64463

Program Counter 0x00BD 189

Instruction Specifier 00000000 STOP

Operand Specifier

(Operand)

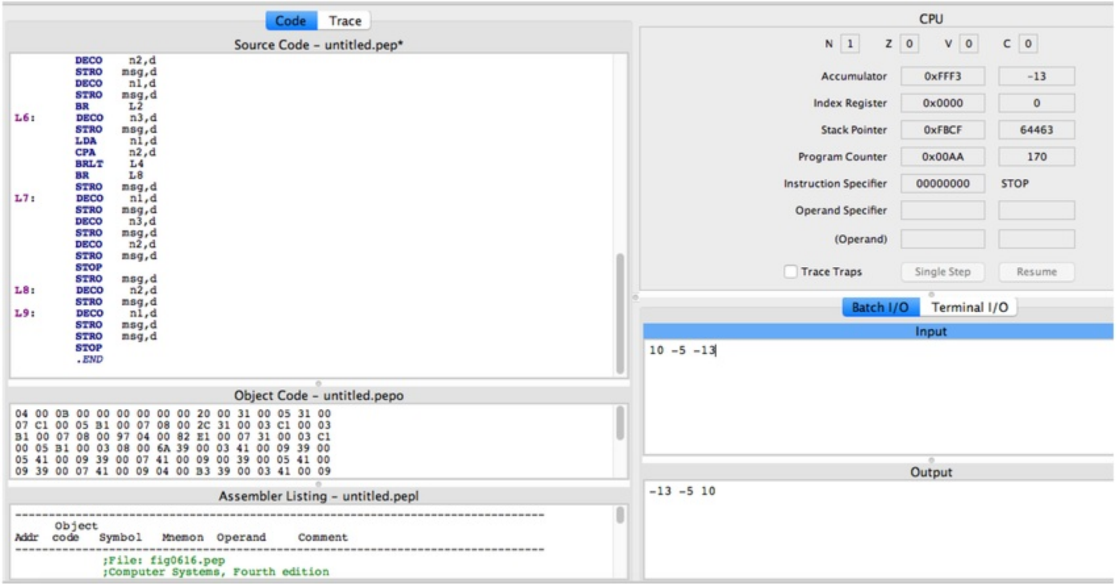
☐ Trace Traps

Input

```
3 -15 25
```

Output

```
-15 3 25
```



Source Code – untitled.pep*

```
         DECO   n2,d
         STRO   msg,d
         DECO   n1,d
         STRO   msg,d
         BR     L2
L6:      DECO   n3,d
         STRO   msg,d
         LDA    n1,d
         CPA    n2,d
         BRLT   L4
         BR     L8
         STRO   msg,d
L7:      DECO   n1,d
         STRO   msg,d
         DECO   n3,d
         STRO   msg,d
         DECO   n2,d
         STRO   msg,d
         STOP
         STRO   msg,d
L8:      DECO   n2,d
         STRO   msg,d
L9:      DECO   n1,d
         STRO   msg,d
         STRO   msg,d
         STOP
         .END
```

Object Code – untitled.pepo

```
04 00 0B 00 00 00 00 00 00 20 00 31 00 05 31 00
07 C1 00 05 B1 00 07 08 00 2C 31 00 03 C1 00 03
B1 00 07 08 00 97 04 00 82 E1 00 07 31 00 03 C1
00 05 B1 00 03 08 00 6A 39 00 03 41 00 09 39 00
05 41 00 09 39 00 07 41 00 09 00 39 00 05 41 00
09 39 00 07 41 00 09 04 00 B3 39 00 03 41 00 09
```

Assembler Listing – untitled.pepl

```
------------------------------------------------------------------------
      Object
Addr  code    Symbol   Mnemon  Operand     Comment
------------------------------------------------------------------------
                ;File: fig0616.pep
                ;Computer Systems, Fourth edition
```

N 1 Z 0 V 0 C 0

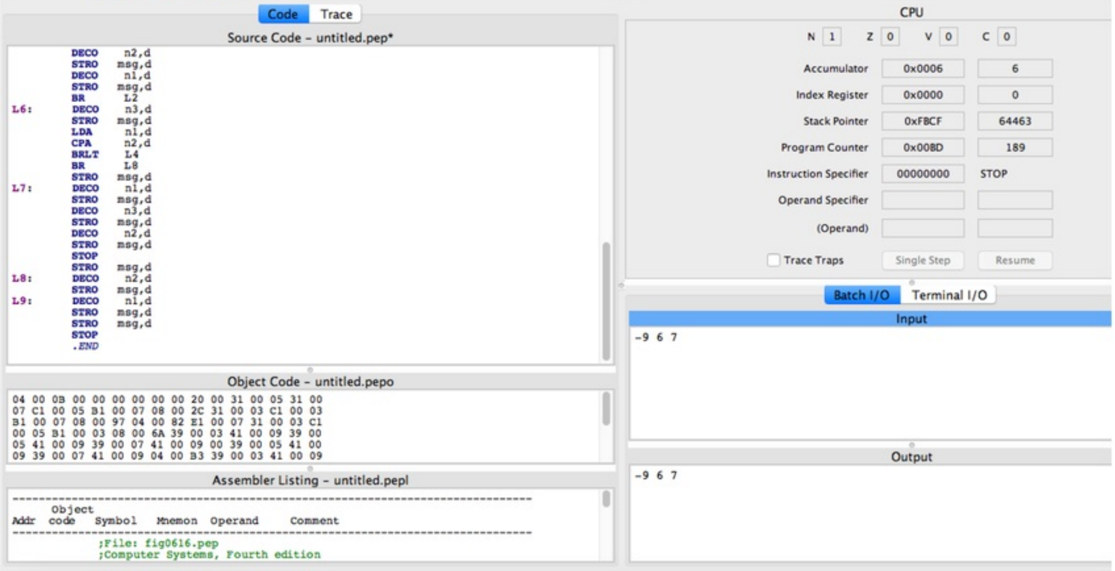
Accumulator 0xFFF3 -13
Index Register 0x0000 0
Stack Pointer 0xFBCF 64463
Program Counter 0x00AA 170
Instruction Specifier 00000000 STOP
Operand Specifier
(Operand)

Input

```
10 -5 -13
```

Output

```
-13 -5 10
```



Source Code – untitled.pep*

```
         DECO   n2,d
         STRO   msg,d
         DECO   n1,d
         STRO   msg,d
         BR     L2
L6:      DECO   n3,d
         STRO   msg,d
         LDA    n1,d
         CPA    n2,d
         BRLT   L4
         BR     L8
         STRO   msg,d
L7:      DECO   n1,d
         STRO   msg,d
         DECO   n3,d
         STRO   msg,d
         DECO   n2,d
         STRO   msg,d
         STOP
         STRO   msg,d
L8:      DECO   n2,d
         STRO   msg,d
L9:      DECO   n1,d
         STRO   msg,d
         STRO   msg,d
         STOP
         .END
```

Object Code – untitled.pepo

```
04 00 0B 00 00 00 00 00 00 20 00 31 00 05 31 00
07 C1 00 05 B1 00 07 08 00 2C 31 00 03 C1 00 03
B1 00 07 08 00 97 04 00 82 E1 00 07 31 00 03 C1
00 05 B1 00 03 08 00 6A 39 00 03 41 00 09 39 00
05 41 00 09 39 00 07 41 00 09 00 39 00 05 41 00
09 39 00 07 41 00 09 04 00 B3 39 00 03 41 00 09
```

Assembler Listing – untitled.pepl

```
------------------------------------------------------------------------
      Object
Addr  code    Symbol   Mnemon  Operand     Comment
------------------------------------------------------------------------
                ;File: fig0616.pep
                ;Computer Systems, Fourth edition
```

N 1 Z 0 V 0 C 0

Accumulator 0x0006 6
Index Register 0x0000 0
Stack Pointer 0xFBCF 64463
Program Counter 0x00BD 189
Instruction Specifier 00000000 STOP
Operand Specifier
(Operand)

Input

```
-9 6 7
```

Output

```
-9 6 7
```

d) Is this spaghetti code or structured code? If it was the other type, would it be easier or harder to modify?

This is spaghetti code because the branching statements occur everywhere in the program. Spaghetti code is much harder to modify.

Problem 2: (14 points) Translate the program below into PEP/8 assembly language

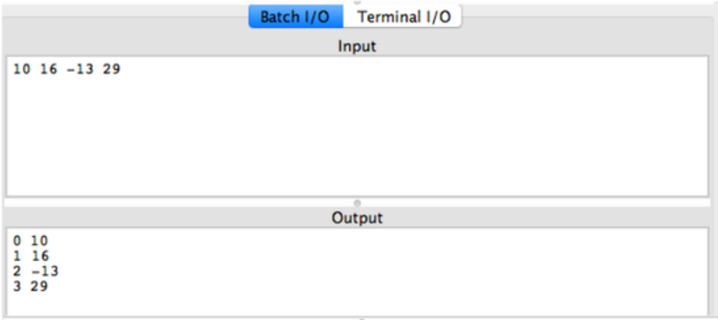
- Start with Assembly code for Fig 6.36 (Use Pep8 help)
- Change to output array in same order as input
- Add function twoVect
- Pass array as parameter as shown in Fig 6.38
- Use trace tags on all variables.

a) Comment lines of the source code to trace the C++ code. Cut & paste the Source Code Listing into your assignment document.

```
BR main
;
;******* toVect (int v[], int n)
v2: .Equate 3 ;formal parameter #2h
n2: .Equate 4 ;formal parameter #2d
j2: .Equate 0 ;formal parameter #2d
toVect: SUBSP 0,i ;allocate #j2 ;WARNING: Number of bytes allocated (0) not equal to number of bytes listed in trace tag (2).
endFor1: LDX 0,i ;for (j = 0
STX j,s
for2: CPX 3,i ; j<3
BRGT endFor2
DECO j,s ; cout << j
CHARO ' ',i ; << ' '
ASLX ; an integer is two bytes
DECO vector,sx ; << vector[j]
CHARO '\n',i ; << endl
LDX j,s ; j++)
ADDX 1,i
STX j,s
BR for2
;
;******* main ()
vector: .EQUATE 2 ;local variable #2d4a
j: .EQUATE 0 ;local variable #2d
main: SUBSP 10,i ;allocate #vector #j
LDX 0,i ;for (j = 0
STX j,s
for1: CPX 4,i ; j < 4
BRGE endFor1
ASLX ; an integer is two bytes
DECI vector,sx ; cin >> vector[j]
LDX j,s ; j++)
ADDX 1,i
STX j,s
BR for1
```

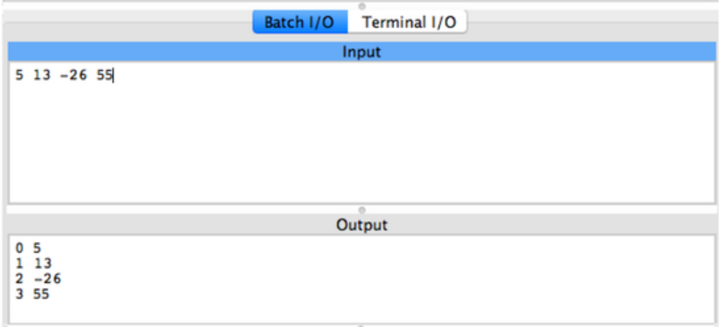
```
CALL toVect ;
endFor2: ADDSP 10,i ;deallocate #j #vector
STOP
.END
```

b) Run for a set of 4 inputs and paste a screen shot of the Output area of PEP/8.



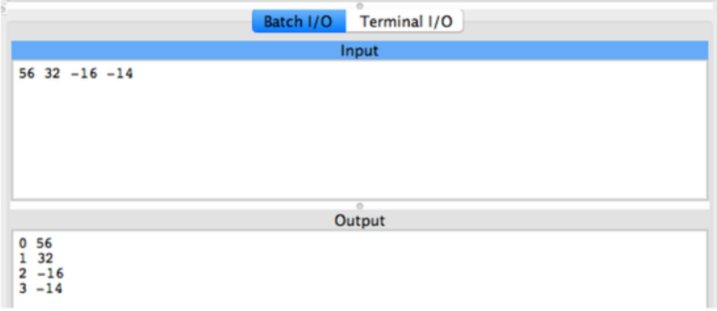
```
10 16 -13 29
```

```
0 10
1 16
2 -13
3 29
```



```
5 13 -26 55
```

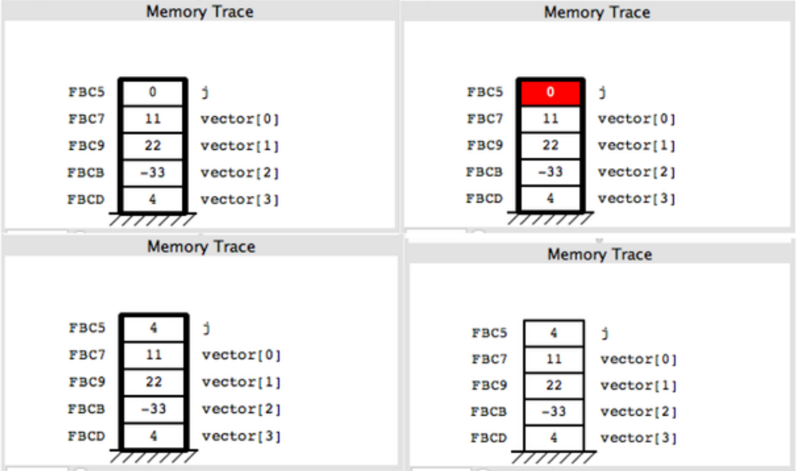
```
0 5
1 13
2 -26
3 55
```



```
56 32 -16 -14
```

```
0 56
1 32
2 -16
3 -14
```

c) Step thru & Cut and paste the memory trace when in the twoVect function



Memory Trace

```
FBC5    0   j
FBC7   11   vector[0]
FBC9   22   vector[1]
FBCB  -33   vector[2]
FBCD    4   vector[3]
```

Memory Trace

```
FBC5    0   j
FBC7   11   vector[0]
FBC9   22   vector[1]
FBCB  -33   vector[2]
FBCD    4   vector[3]
```

Memory Trace

```
FBC5    4   j
FBC7   11   vector[0]
FBC9   22   vector[1]
FBCB  -33   vector[2]
FBCD    4   vector[3]
```

Memory Trace

```
FBC5    4   j
FBC7   11   vector[0]
FBC9   22   vector[1]
FBCB  -33   vector[2]
FBCD    4   vector[3]
```

Problem 3: (14 points)
Translate the program below into PEP/8 assembly language

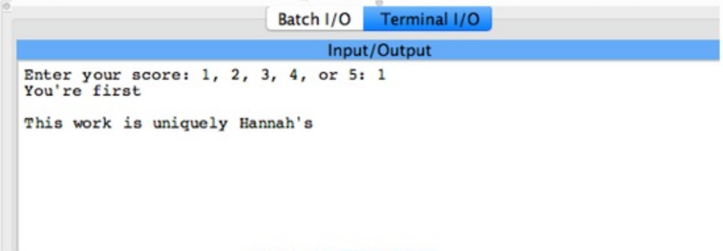
- Use a jump table to implement the switch statement.
- Use trace tags on all variables.
- For invalid scores, output should be the same as the C++ program.
- Add something to the output that makes this program uniquely yours.
- The variable finish needs to be local.
- This is similar to Fig 6.40.

a) Comment lines of the source code to trace the C++ code. Cut & paste the Source Code Listing into your assignment document.

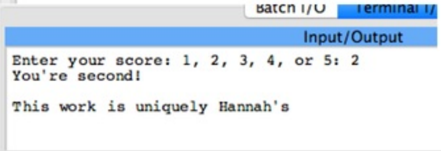
```
BR main
;
;******* main ()
score: .EQUATE 0 ;local variable #2d
main: SUBSP 2,i ;allocate #score
STRO msgIn,d ;cout << "Enter your score: 1, 2, 3, 4, or 5: "
DECI score,s ;cin >> score
LDX score,s ;switch (score)
LDA score,s ;Load into the accumulator
SUBA 1,i ;subtract 1
BRLT errorC,i ;Branch less than 0
LDA score,s ;load score to the accumulator register
SUBA 5,i ;subtract 5
BRGT errorC,i ;Branch greater than 0
ASLX ;addresses occupy two bytes
BR guessJT,x
guessJT: .ADDRSS case0
```

```
.ADDRSS case1
.ADDRSS case2
.ADDRSS case3
.ADDRSS case4
.ADDRSS case5
case0: BR endCase ;break
case1: STRO msg0,d ;cout << "You're first!"
BR endCase ;break
case2: STRO msg1,d ;cout << "You're second!"
BR endCase ;break
case3: STRO msg2,d ;cout << "You're not first or second"
BR endCase ;break
case4: STRO msg2,d ;cout << "You're not first or second"
BR endCase
case5: STRO msg2,d ;cout << "You're not first or second"
BR endCase
errorC: STRO msg3,d ;cout << "You weren't even competing"
endCase: STRO msg4,d
ADDSP 2,i ;deallocate #guess ;WARNING: guess not specified in .EQUATE.
STOP
msgIn: .ASCII "Enter your score: 1, 2, 3, 4, or 5: \x00"
msg0: .ASCII "You're first!\x00"
msg1: .ASCII "You're second!\x00"
msg2: .ASCII "You're not first or second\x00"
msg3: .ASCII "You weren't even competing\x00"
msg4: .ASCII "\n\nThis work is uniquely Hannah's\x00"
.END
```

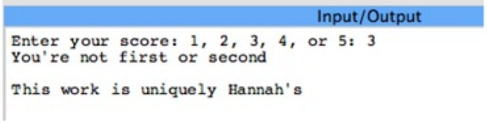
b) Run for each score and paste a screen shot of each of the PEP/8 Output area.



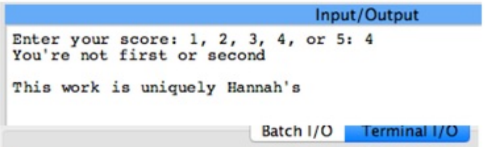
```
Enter your score: 1, 2, 3, 4, or 5: 1
You're first

This work is uniquely Hannah's
```



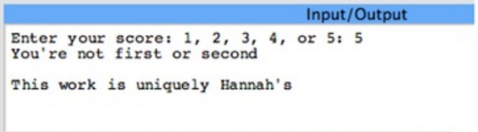
```
Enter your score: 1, 2, 3, 4, or 5: 2
You're second!

This work is uniquely Hannah's
```



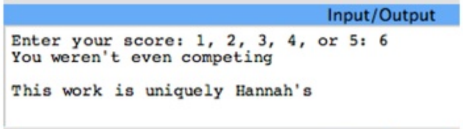
```
Enter your score: 1, 2, 3, 4, or 5: 3
You're not first or second

This work is uniquely Hannah's
```



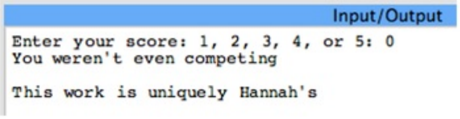
```
Enter your score: 1, 2, 3, 4, or 5: 4
You're not first or second

This work is uniquely Hannah's
```



```
Enter your score: 1, 2, 3, 4, or 5: 5
You're not first or second

This work is uniquely Hannah's
```

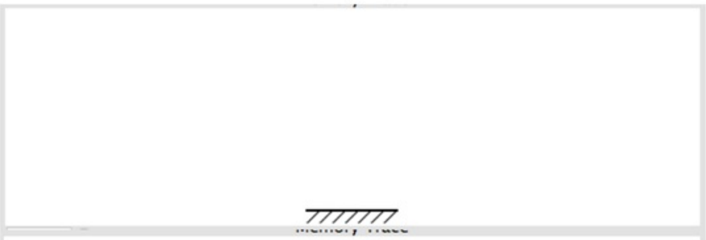


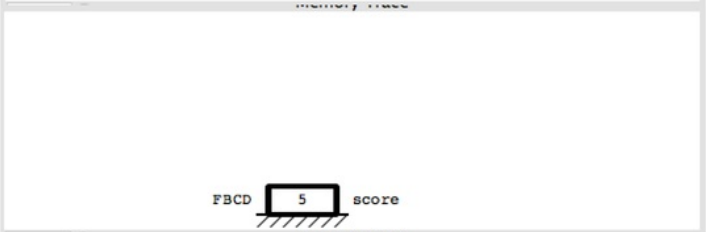
```
Enter your score: 1, 2, 3, 4, or 5: 6
You weren't even competing

This work is uniquely Hannah's
```

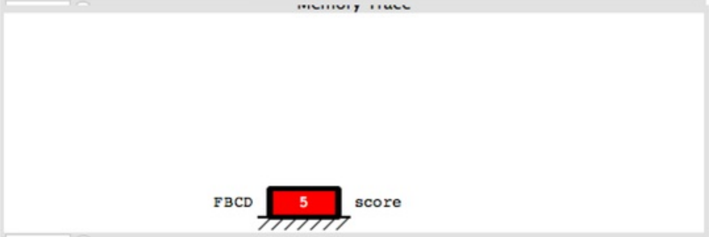


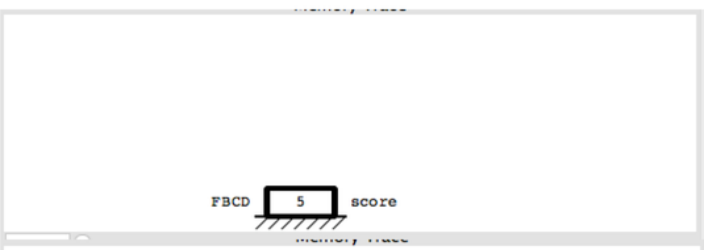
```
Enter your score: 1, 2, 3, 4, or 5: 0
You weren't even competing

This work is uniquely Hannah's
```

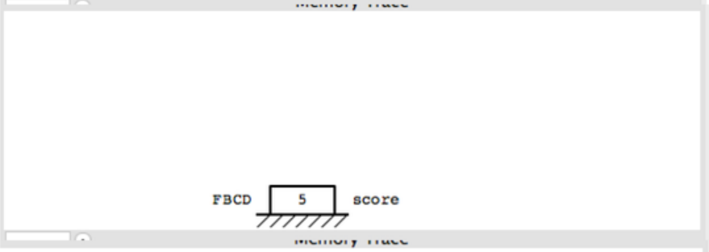
d) Step thru & Cut and paste the memory trace at any point.








Memory Trace
FBCD
5
score

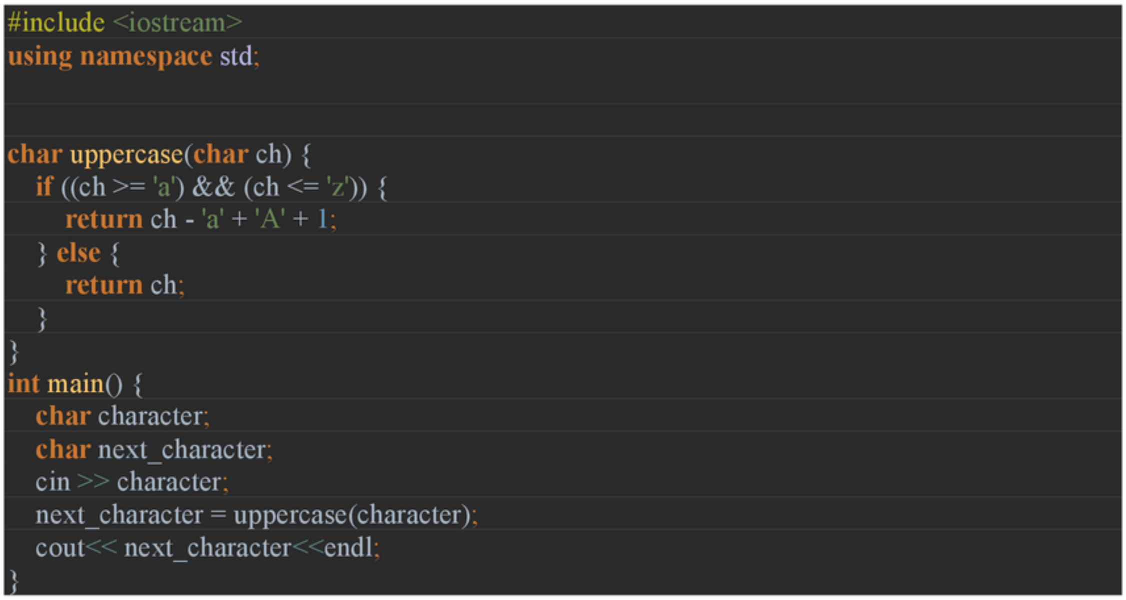

Memory Trace
FBCD
5
score

Memory Trace

Problem 4: (14 points)
Write a C++ program that inputs a lower case character, converts it to an upper case character using the function below and increments it to the next character (i.e. B will be changed to C). If Z is entered it should be changed to A. If a non-letter character is entered, it should not be changed.

- A character that is not a letter should be returned as is.
- Character variables will need character trace tags.
- Hint: characters only use one byte of storage and should be manipulated with byte instructions.
- Add something to the output that makes this program uniquely yours.
- Then translate it to Assembly language.

a) Cut and paste you C++ Source Code into your assignment document.



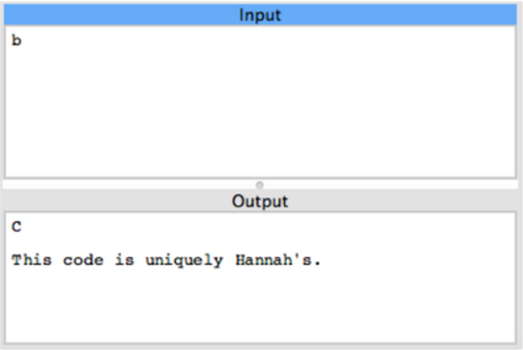
```
#include <iostream>
using namespace std;


char uppercase(char ch) {
   if ((ch >= 'a') && (ch <= 'z')) {
      return ch - 'a' + 'A' + 1;
   } else {
      return ch;
   }
}
int main() {
   char character;
   char next_character;
   cin >> character;
   next_character = uppercase(character);
   cout<< next_character<<endl;
}
```

b.) Comment lines of the source code to trace the C++ code. Cut & paste the Assembly Source Code Listing into your assignment document.

```
BR main
;**** main()
j: .EQUATE 1 ;local variable #1c
main: SUBSP 1,i ;allocate #j
CHARI j,d ;input character
LDBYTEA j,d ; load byte to A
CPA 'a',i ; compare byte to A
BRLT else ; branch if less than
CPA 'z',i ; compare byte to A
BRGT else ; branch if greater to A
```

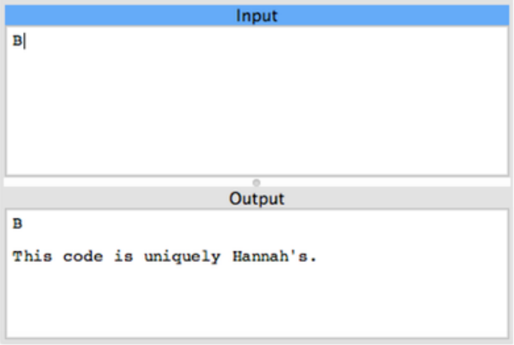
```
CPA 'z',i ; compare bypte to A
BREQ equal ; branch if equal to
SUBA 0x0020,i ; convert to upper case
ADDA 0x0001,i ; increment by 1
STBYTEA j,d ; store j in A
CHARO j,d ; output j
STRO msg,d ; code is unique
BR endIf ; branch to endIf
else: CHARO j,d ; else print the same character
STRO msg,d ; code is unique
endIf: ADDSP 1,i ; deallocate #j
STOP
equal: CHARO 'A',i ; else print A
STRO msg,d ; code is unique
msg: .ASCII "\n\nThis code is uniquely Hannah's. \x00"
.END
```

c.) Run for 3 inputs: one uppercase, one lowercase, & one non-letter and paste a screen shot of each in the Output area of the PEP/8.



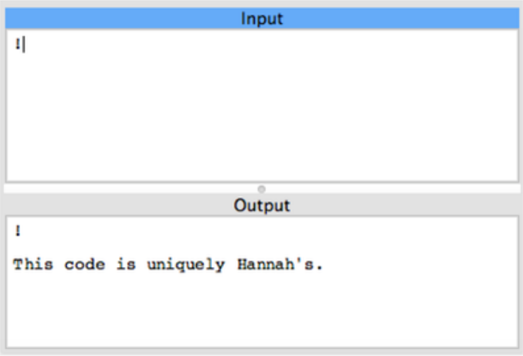
```
b
```

```
C

This code is uniquely Hannah's.
```



```
B
```

```
B

This code is uniquely Hannah's.
```

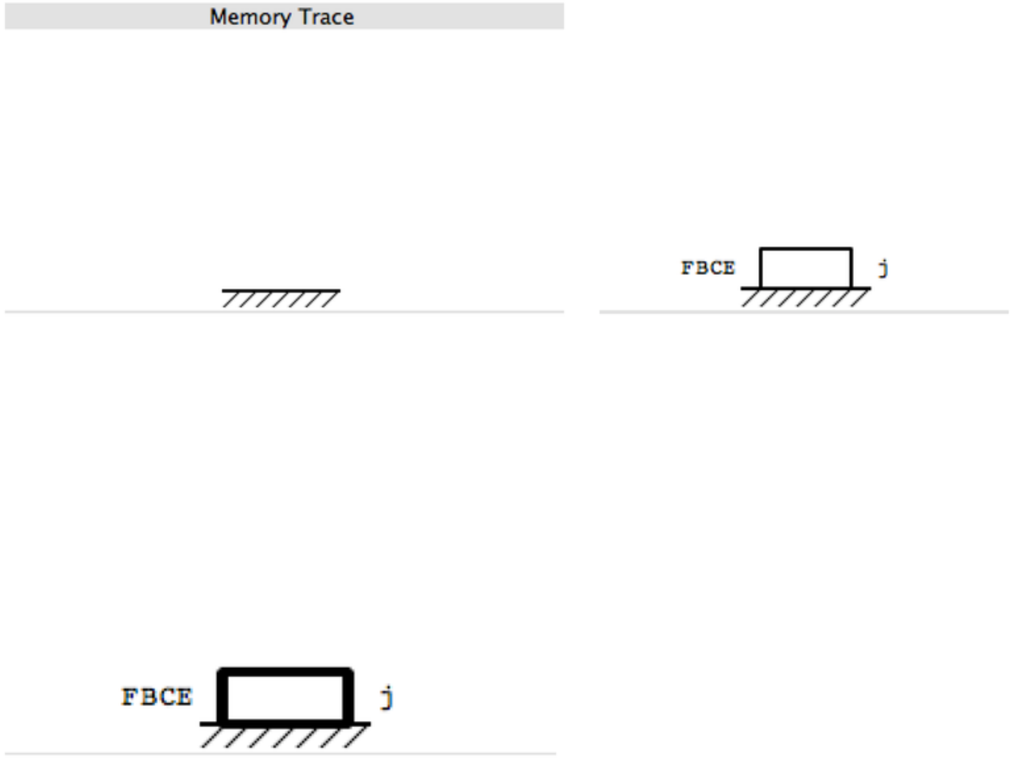


```
!
```

```
!

This code is uniquely Hannah's.
```

d) Step thru & Cut and paste the memory trace at a point when in uppercase subroutine.

```
char uppercase (char ch) { if ((ch >= 'a') && (ch <= 'z')) { return ch - 'a' + 'A'; } else { return ch;
} }
```



hroac2HW4

GRADEMARK REPORT

FINAL GRADE

47/50

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

Comment 1

Double and output in order

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

RUBRIC: SPR2015-HW4 47 / 50

1A 2 / 2

FULL CREDIT
(2)

MINUS 1
(1)

MINUS 2
(0)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

1B 2 / 2

FULL CREDIT
(2)

MINUS 1
(1)

MINUS 2
(0)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

1C 2 / 2

FULL CREDIT
(2)

MINUS 1
(1)

MINUS 2
(0)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

1D 2 / 2

FULL CREDIT
(2)

MINUS 1
(1)

MINUS 2
(0)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

2A 4 / 7

FULL CREDIT
(7)

MINUS 1
(6)

MINUS 2
(5)

MINUS 3
(4)

MINUS 4
(3)

MINUS 5
(2)

MINUS 6
(1)

NO CREDIT
(0)

2B 4 / 4

FULL CREDIT
(4)

MINUS 1
(3)

MINUS 2
(2)

MINUS 3
(1)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

2C 3 / 3

FULL CREDIT
(3)

MINUS 1
(2)

MINUS 2
(1)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

3A 7 / 7

FULL CREDIT
(7)

MINUS 1
(6)

MINUS 2
(5)

MINUS 3
(4)

MINUS 4
(3)

MINUS 5
(2)

MINUS 6
(1)

NO CREDIT
(0)

3B 4 / 4

FULL CREDIT

(4)

MINUS 1
(3)

MINUS 2
(2)

MINUS 3
(1)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

3C	3 / 3

FULL CREDIT
(3)

MINUS 1
(2)

MINUS 2
(1)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

4A	2 / 2

FULL CREDIT
(2)

MINUS 1
(1)

MINUS 2
(0)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

4B 7 / 7

FULL CREDIT
(7)

MINUS 1
(6)

MINUS 2
(5)

MINUS 3
(4)

MINUS 4
(3)

MINUS 5
(2)

MINUS 6
(1)

NO CREDIT
(0)

4C 3 / 3

FULL CREDIT
(3)

MINUS 1

(2)

MINUS 2
(1)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)

4D 2 / 2

FULL CREDIT
(2)

MINUS 1
(1)

MINUS 2
(0)

MINUS 3
(0)

MINUS 4
(0)

MINUS 5
(0)

MINUS 6
(0)

NO CREDIT
(0)