Keep this test booklet when you are done. **Closed book. No notes.** Work strictly from memory. **No calculators. Scratch paper okay.**

Section I: Record all answers on the bubble sheet.

On the following printf questions you are given a list of inputs. For each problem line determine which printf statement created the accompanying outputs. (␣ means space.)

Which of these printf statements created the outputs shown for each problem below? (`x` is `int x;`)

(A) `printf("␣␣␣%-␣5d␣",x);`
(B) `printf("␣␣␣%-5d␣",x);`
(C) `printf("␣%␣08d",x);`
(D) `printf("␣%+5d␣␣␣",x);`
(E) `printf("␣%-␣4d␣␣␣␣",x);`
(F) `printf("␣%-␣7d␣",x);`
(G) `printf("%+09d",x);`
(H) `printf("%-+9d",x);`
(I) `printf("%-9d",x);`

inputs:	3	-4	1961770358	-2009563461
1/2p.	␣␣␣␣3␣␣␣␣	␣␣␣-4␣␣␣␣	␣␣␣␣1961770358␣	␣␣␣-2009563461␣
2/2p.	3␣␣␣␣␣␣␣␣	-4␣␣␣␣␣␣␣	1961770358	-2009563461
3/2p.	␣␣3␣␣␣␣␣␣	␣-4␣␣␣␣␣␣	␣␣1961770358␣	␣-2009563461␣
4/2p.	␣␣0000003	␣-0000004	␣␣1961770358	␣-2009563461

Which of these printf statements created the outputs shown for each problem below? (`x` is `char * x;`)

(A) `printf("␣␣␣␣␣%s",x);`
(B) `printf("␣␣␣%1s␣",x);`
(C) `printf("␣␣%1s␣␣",x);`
(D) `printf("␣␣%3s",x);`
(E) `printf("␣%-3s␣",x);`
(F) `printf("␣%2s␣␣",x);`
(G) `printf("␣%4s",x);`
(H) `printf("␣%s␣␣␣␣",x);`
(I) `printf("%-5s",x);`

inputs:	""	"d"	"qd"	"vpkd"	"phhpfb"	"hxhvplpk"
5/2p.	␣␣␣␣␣	␣␣␣␣␣d	␣␣␣␣␣qd	␣␣␣␣␣vpkd	␣␣␣␣␣phhpfb	␣␣␣␣␣hxhvplpk
6/2p.	␣␣␣␣␣	␣d␣␣␣	␣qd␣␣	␣vpkd␣	␣phhpfb␣	␣hxhvplpk␣
7/2p.	␣␣␣␣␣	␣␣d␣␣	␣qd␣␣	␣vpkd␣␣	␣phhpfb␣␣	␣hxhvplpk␣␣
8/2p.	␣␣␣␣␣	d␣␣␣␣	qd␣␣␣	vpkd␣	phhpfb	hxhvplpk

Which of these printf statements created the outputs shown for each problem below? (`x` is `double x;`)

(A) `printf("␣␣␣␣%010.0f",x);`
(B) `printf("␣␣␣%10.2f␣",x);`
(C) `printf("␣␣%+10f␣␣",x);`
(D) `printf("␣%+010.2f␣␣␣",x);`
(E) `printf("␣%+13.2f",x);`
(F) `printf("␣%0+11.0f␣␣",x);`
(G) `printf("␣%013.0f",x);`
(H) `printf("%+13f␣",x);`
(I) `printf("%+14.0f",x);`

inputs:	6	-5.64	-4.1515	-169788.317396
9/2p.	␣␣␣␣␣␣␣␣␣␣␣␣+6	␣␣␣␣␣␣␣␣␣␣␣␣-6	␣␣␣␣␣␣␣␣␣␣␣␣-4	␣␣␣␣␣␣␣-169788
10/2p.	␣␣␣␣␣␣␣␣␣+6.00	␣␣␣␣␣␣␣␣␣-5.64	␣␣␣␣␣␣␣␣␣-4.15	␣␣␣␣-169788.32
11/2p.	␣␣␣␣0000000006	␣␣␣␣-000000006	␣␣␣␣-000000004	␣␣␣␣-000169788
12/2p.	␣␣␣␣+6.000000␣	␣␣␣␣-5.640000␣	␣␣␣␣-4.151500␣	-169788.317396␣

Precedence: What is the value of each expression? Mark (I) for error, (J) for none of the above.

13/1p.	`5+7/1>=6!=1`	(A) -61	(B) -58	(C) 0	(D) 2	(E) 5	(F) 6	(G) 12	(H) 83
14/1p.	`8*8/8+8/3`	(A) -58	(B) 0	(C) 1	(D) 5	(E) 6	(F) 8	(G) 10	(H) 24
15/1p.	`2%5||5||5-4`	(A) -28	(B) -4	(C) -3	(D) -1	(E) 0	(F) 1	(G) 6	(H) 85
16/1p.	`9+7!=5&&0*5`	(A) 1	(B) 5	(C) 9	(D) 10	(E) 14	(F) 45	(G) 50	(H) 86
17/1p.	`2%5-9/9/7`	(A) -51	(B) -7	(C) -2	(D) -1	(E) 0	(F) 2	(G) 26	(H) 28
18/1p.	`5%0-2-7-3`	(A) -41	(B) -11	(C) -7	(D) -3	(E) -1	(F) 1	(G) 5	(H) 77
19/1p.	`5/1-5%9*2`	(A) -10	(B) -4	(C) -2	(D) -1	(E) 0	(F) 2	(G) 14	(H) 72
20/1p.	`2+8==3&&8+3`	(A) -81	(B) -17	(C) -4	(D) 0	(E) 1	(F) 2	(G) 3	(H) 4
21/1p.	`1*7/4-8-3`	(A) -63	(B) -56	(C) -45	(D) -10	(E) -7	(F) -5	(G) -4	(H) -1
22/1p.	`7/4/2*0-5`	(A) -98	(B) -15	(C) -7	(D) -5	(E) -2	(F) 12	(G) 29	(H) 57
23/1p.	`8-6/1%4+3`	(A) -1	(B) 0	(C) 2	(D) 3	(E) 5	(F) 7	(G) 9	(H) 44
24/1p.	`5-7/9/7*7`	(A) -45	(B) -14	(C) -7	(D) -1	(E) 0	(F) 5	(G) 35	(H) 38
25/1p.	`6%6-2%6-1`	(A) -56	(B) -28	(C) -3	(D) -2	(E) 0	(F) 1	(G) 2	(H) 65
26/1p.	`8*1-0-9/9`	(A) -54	(B) -1	(C) 1	(D) 7	(E) 8	(F) 9	(G) 16	(H) 21
27/1p.	`9<=9<=2*6+8`	(A) -46	(B) 0	(C) 1	(D) 9	(E) 14	(F) 21	(G) 39	(H) 75
28/1p.	`6*5+1+8/9`	(A) -90	(B) 4	(C) 6	(D) 9	(E) 20	(F) 36	(G) 53	(H) 66
29/1p.	`2*7>=5<=8-5`	(A) -27	(B) -8	(C) -4	(D) -3	(E) 1	(F) 2	(G) 16	(H) 63
30/1p.	`8/2-0%5-6`	(A) -53	(B) -27	(C) 0	(D) 1	(E) 4	(F) 10	(G) 11	(H) 63
31/1p.	`3%8%7-8/5`	(A) -85	(B) -46	(C) -13	(D) -2	(E) 0	(F) 1	(G) 2	(H) 3

How many times does the body of the loop execute? (Mark 9 if 9 or more.)

32/2p. `int v; for( v=-3; v>-3; v-- ) body;`

33/2p. `int b=6; do body; while( b++ <= 8 );`

34/2p. `int e=1; while( --e != -4 ) body;`

35/2p. `int e; for( e=3; e!=8; ++e ) body;`

36/2p. `int f=7; while( f++ < 10 ) body;`

37/2p. `int i; for( i=4; i>2; i-- ) body;`

38/2p. `int p=10; while( ++p < 13 ) body;`

39/2p. `int k=-8; while( k++ <= -9 ) body;`

40/3p. Give a tight big-oh $\Theta()$ bound on the running time T(n) of this program.
Assume `atoi`, `simpleStatement`, and `simpleCompare` each run in $\Theta(1)$ time.
(A) $n^2\sqrt{n}$ (C) n^2 (E) $n\sqrt{n}$ (G) n (I) $\sqrt{n}$
(B) $n^2 \lg n$ (D) $n\sqrt{n}\lg n$ (F) $n \lg n$ (H) $\sqrt{n}\lg n$ (J) $\lg n$

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      b = 1; do {
         simpleStatement;
      b++; } while ( b < n );
   } else {
      simpleStatement;
   }
   return 0; }
```

41/3p. Give a tight big-oh $\Theta()$ bound on the running time T(n) of this program.
Assume `atoi`, `simpleStatement`, and `simpleCompare` each run in $\Theta(1)$ time.
(A) $n^2\sqrt{n}$ (C) $n\sqrt{n}\lg n$ (E) n (G) $\sqrt{n}$ (I) $\lg n$
(B) n^2 (D) $n\sqrt{n}$ (F) $\sqrt{n}\lg n$ (H) $\lg^2 n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      j = n; while ( j > 1 ) {
         g = 1; do {
            simpleStatement;
         g *= 2; } while ( g < n );
      j /= 2; }
   } else {
      simpleStatement;
   }
   return 0; }
```

42/3p. Give a tight big-oh $\Theta()$ bound on the running time T(n) of this program.
Assume `atoi`, `simpleStatement`, and `simpleCompare` each run in $\Theta(1)$ time.
(A) n^2 (C) $n\sqrt{n}$ (E) n (G) $\sqrt{n}$ (I) $\lg n$
(B) $n\sqrt{n}\lg n$ (D) $n \lg n$ (F) $\sqrt{n}\lg n$ (H) $\lg^2 n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   for ( a = n ; a > 1 ; a-- ) {
      if ( simpleCompare ) {
         if ( simpleCompare ) {
            simpleStatement;
         }
      } else {
         simpleStatement;
      }
   }
   return 0; }
```

43/5p. Give a tight big-oh $\Theta()$ bound on the running time T(n) of this program.
Assume `atoi`, `simpleStatement`, and `simpleCompare` each run in $\Theta(1)$ time.
(A) $n^2\sqrt{n}$ (C) $n\sqrt{n}\lg n$ (E) n (G) $\sqrt{n}$ (I) $\lg n$
(B) n^2 (D) $n\sqrt{n}$ (F) $\sqrt{n}\lg n$ (H) $\lg^2 n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      b = n; do {
         if ( simpleCompare ) {
            if ( simpleCompare ) {
               simpleStatement;
            }
         } else {
            simpleStatement;
         }
      b /= 3; } while ( b > 1 );
   } else {
      c = n; while ( c > 1 ) {
         simpleStatement;
      c /= 2; }
   }
   return 0; }
```

44/5p. Give a tight big-oh $\Theta()$ bound on the running time T(n) of this program.
Assume `atoi`, `simpleStatement`, and `simpleCompare` each run in $\Theta(1)$ time.
(A) $n^2\sqrt{n}$ (C) n^2 (E) $\sqrt{n}\lg^2 n$ (G) $\sqrt{n}$ (I) $\lg n$
(B) $n^2\lg n$ (D) $n\lg n$ (F) $\sqrt{n}\lg n$ (H) $\lg^3 n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   for ( f = n ; f > 1 ; f -= 2 ) {
      if ( simpleCompare ) {
         a = 1; while ( a < n ) {
            if ( simpleCompare ) {
               g = n; while ( g > 1 ) {
                  simpleStatement;
               g -= 10; }
            } else {
               simpleStatement;
            }
         a *= 2; }
      } else {
         if ( simpleCompare ) {
            simpleStatement;
         }
      }
   }
   return 0; }
```

45/5p. Give a tight big-oh $\Theta()$ bound on the running time T(n) of this program.
Assume `atoi`, `simpleStatement`, and `simpleCompare` each run in $\Theta(1)$ time.

(A) $n^2\sqrt{n}$ (C) n^2 (E) $n\sqrt{n}$ (G) n (I) $\lg n$
(B) $n^2 \lg n$ (D) $n\sqrt{n}\lg n$ (F) $n \lg n$ (H) $\sqrt{n}\lg n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      if ( simpleCompare ) {
         if ( simpleCompare ) {
            g = n; do {
               if ( simpleCompare ) {
                  simpleStatement;
               } else {
                  simpleStatement;
               }
            g -= 5; } while ( g > 1 );
         } else {
            if ( simpleCompare ) {
               simpleStatement;
            } else {
               simpleStatement;
            }
         }
      } else {
         if ( simpleCompare ) {
            h = 1; while ( h * h < n ) {
               simpleStatement;
            h += 2; }
         } else {
            simpleStatement;
         }
      }
   } else {
      if ( simpleCompare ) {
         for ( j = 1 ; j < n ; j *= 5 ) {
            b = 1; while ( b < n ) {
               if ( simpleCompare ) {
                  simpleStatement;
               }
            b *= 3; }
         }
      } else {
         for ( d = n ; d > 1 ; d /= 5 ) {
            simpleStatement;
         }
      }
   }
   return 0; }
```

Matching: Which Perl regular expression commands have what meaning? (If no match mark J.)

(A) `C-X` (B) `\A` (C) `\a` (D) `\cX` (E) `\f` (F) `\i` (G) `\n` (H) `\t` (I) `ff`

46/1p. tab
47/1p. alarm (alert)
48/1p. form feed
49/1p. control X
50/1p. carriage return

True or False: does the string match the regular expression?

51/1p. Does the string "`xbxbxy`" match the regular expression "`xb*|xy`"?

52/1p. Does the empty string match the regular expression "`u*|bh`"?

53/1p. Does the string "`nzzu`" match the regular expression "`nz+u?`"?

54/1p. Does the empty string match the regular expression "`(fy*|cs)?`"?

55/1p. Does the string "`fa`" match the regular expression "`fa|n?`"?

56/1p. Does the string "`qnbhh`" match the regular expression "`(qn+bh)*`"?

57/1p. Does the string "`zhhshud`" match the regular expression "`zh+|(sh+ud)*`"?

58/1p. Does the string "`tfctfcwb`" match the regular expression "`(tf?c)+wb*`"?

59/1p. Does the string "`qbb`" match the regular expression "`(qb)?|qr(u*sx)*`"?

60/1p. Does the string "`yqqanananananyy`" match the regular expression "`(yq)*((an)?)?y?`"?

61/1p. Does the string "`kqwwrr`" match the regular expression "`kq+(wa?|zh)+wr*`"?

62/1p. Does the string "`rb`" match the regular expression "`(df)*|rb|((bk)?)?|yq?`"?

Section II: Non-Bubble Answers: Write neatly. Use the paper provided. Answer on the front ONLY of each page. On the upper-right corner of the back, lightly write your 7-digit BYUH student id number. Do NOT write your NAME anywhere. (This aids fairness in grading.) Leave margins the same size as on this page. Sloppy/non-compliant work will be penalized.

Write each program on a separate sheet of paper. Write the program number in the upper left corner and circle it.

63/10p. Write this program: Prompt for and accept two integers, a and b. Print their quotient (a divided by b).

64/10p. Write this program: Accept a list of one or more integers from the command line (argv). Print the largest (max).

65/10p. Write this program: Given n on the command line, print a triangle of stars, 1 on top, n at the bottom.

66/10p. Write this program: Write a subroutine that, given a list of words, returns the same list with duplicates removed. Specifically, return the first occurrance of each word. Sample input: a b a x c c a a d b, output: a b x c d.

67/10p. Write this program: Adding time. Prompt for and accept a start time and a running time in HH:MM:SS format. Print the ending time. Example: 1:39:44 + 2:11:31 = 3:51:15.

Total points 150.

Answer Key (points per line)

1 (2).	A	32 (2).	0
2 (2).	I	33 (2).	4
3 (2).	F	34 (2).	4
4 (2).	C	35 (2).	5
5 (2).	A	36 (2).	3
6 (2).	E	37 (2).	2
7 (2).	F	38 (2).	2
8 (2).	I	39 (2).	0
9 (2).	I	40 (3).	G (n)
10 (2).	E	41 (3).	H ($\lg^2 n$)
11 (2).	A	42 (3).	E (n)
12 (2).	H	43 (5).	I ($\lg n$)
13 (1).	C (0)	44 (5).	B ($n^2 \lg n$)
14 (1).	G (10)	45 (5).	G (n)
15 (1).	F (1)	46 (1).	H
16 (1).	J (0)	47 (1).	C
17 (1).	F (2)	48 (1).	E
18 (1).	I (error)	49 (1).	D
19 (1).	J (-5)	50 (1).	J
20 (1).	D (0)	51 (1).	false
21 (1).	D (-10)	52 (1).	true
22 (1).	D (-5)	53 (1).	true
23 (1).	G (9)	54 (1).	true
24 (1).	F (5)	55 (1).	true
25 (1).	C (-3)	56 (1).	false
26 (1).	D (7)	57 (1).	false
27 (1).	C (1)	58 (1).	true
28 (1).	J (31)	59 (1).	false
29 (1).	E (1)	60 (1).	false
30 (1).	J (-2)	61 (1).	true
31 (1).	G (2)	62 (1).	true

Total points 150.