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("␣␣%+d␣␣␣␣",x);` (D) `printf("␣%-2d␣␣␣␣",x);` (G) `printf("%+-3d␣␣␣␣",x);`
(B) `printf("␣%+3d␣␣␣",x);` (E) `printf("␣%0+6d",x);` (H) `printf("%+04d␣␣␣",x);`
(C) `printf("␣%+4d␣␣",x);` (F) `printf("%␣-6d␣",x);` (I) `printf("%-5d␣␣",x);`

inputs:	9	-4	1518115413	-1973386948
1/2p.	`+9␣␣␣␣␣`	`-4␣␣␣␣␣`	`+1518115413␣␣␣␣`	`-1973386948␣␣␣␣`
2/2p.	`␣␣+9␣␣␣`	`␣␣-4␣␣␣`	`␣+1518115413␣␣␣`	`␣-1973386948␣␣␣`
3/2p.	`␣+00009`	`␣-00004`	`␣+1518115413`	`␣-1973386948`
4/2p.	`␣9␣␣␣␣␣`	`␣-4␣␣␣␣`	`␣1518115413␣␣␣␣`	`␣-1973386948␣␣␣␣`

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

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

inputs:	""	"g"	"wd"	"wwxp"	"lbwlqk"	"fgjwzfyg"
5/2p.	`␣␣␣␣␣`	`g␣␣␣␣`	`wd␣␣␣`	`wwxp␣`	`lbwlqk`	`fgjwzfyg`
6/2p.	`␣␣␣␣␣`	`␣g␣␣␣`	`␣wd␣␣`	`␣wwxp`	`␣lbwlqk`	`␣fgjwzfyg`
7/2p.	`␣␣␣␣␣`	`␣␣␣g␣␣`	`␣␣␣wd␣␣`	`␣␣␣wwxp␣␣`	`␣␣␣lbwlqk␣␣`	`␣␣␣fgjwzfyg␣␣`
8/2p.	`␣␣␣␣␣`	`␣␣␣␣g`	`␣␣␣wd`	`␣wwxp`	`␣lbwlqk`	`␣fgjwzfyg`

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

(A) `printf("␣␣%+10.2f␣␣",x);` (D) `printf("␣%+013.6f",x);` (G) `printf("%+14.6f",x);`
(B) `printf("␣␣%11.4f␣",x);` (E) `printf("␣%010.4f␣␣␣",x);` (H) `printf("%011.2f␣␣␣",x);`
(C) `printf("␣%+013.2f",x);` (F) `printf("%+12.6f␣␣",x);` (I) `printf("%012.6f␣␣",x);`

inputs:	0	1.62	-7.1979	-56657.887886
9/2p.	`␣␣␣␣␣␣␣␣+0.00␣␣`	`␣␣␣␣␣␣␣␣+1.62␣␣`	`␣␣␣␣␣␣␣␣-7.20␣␣`	`␣␣␣-56657.89␣␣`
10/2p.	`␣␣␣␣␣␣+0.000000`	`␣␣␣␣␣␣+1.620000`	`␣␣␣␣␣␣-7.197900`	`␣-56657.887886`
11/2p.	`␣+000000000.00`	`␣+000000001.62`	`␣-000000007.20`	`␣-000056657.89`
12/2p.	`␣00000.0000␣␣␣`	`␣00001.6200␣␣␣`	`␣-0007.1979␣␣␣`	`␣-56657.8879␣␣␣`

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

13/1p.	`2<9&&2+4-7`	(A) -30	(B) -8	(C) -7	(D) -6	(E) -3	(F) -2	(G) 0	(H) 61
14/1p.	`2%4!=2||4-7`	(A) -71	(B) -56	(C) -7	(D) -6	(E) -4	(F) 0	(G) 1	(H) 89
15/1p.	`8%8/4-5+8`	(A) -49	(B) -13	(C) -3	(D) 0	(E) 2	(F) 3	(G) 7	(H) 8
16/1p.	`8*6||8&&2+3`	(A) -89	(B) -88	(C) -83	(D) -49	(E) 1	(F) 4	(G) 11	(H) 32
17/1p.	`2%7-0-3-5`	(A) -31	(B) -6	(C) -3	(D) 0	(E) 2	(F) 4	(G) 16	(H) 47
18/1p.	`5/4/4-6/7`	(A) -92	(B) -70	(C) -38	(D) -5	(E) 0	(F) 2	(G) 5	(H) 93
19/1p.	`8%4%7*3-5`	(A) -23	(B) -12	(C) -5	(D) -2	(E) 0	(F) 3	(G) 23	(H) 29
20/1p.	`4%8%7%9-2`	(A) -28	(B) -20	(C) 0	(D) 1	(E) 2	(F) 85	(G) 89	(H) 92
21/1p.	`7-6/5-7/8`	(A) -87	(B) -29	(C) -1	(D) 0	(E) 1	(F) 6	(G) 8	(H) 13
22/1p.	`6+1*9%2+7`	(A) -23	(B) 0	(C) 6	(D) 8	(E) 14	(F) 40	(G) 43	(H) 48
23/1p.	`8*7+4/6/3`	(A) 0	(B) 3	(C) 4	(D) 18	(E) 40	(F) 44	(G) 56	(H) 72
24/1p.	`4*6%9%8-2`	(A) -86	(B) -83	(C) -12	(D) -8	(E) -2	(F) 0	(G) 4	(H) 16
25/1p.	`6/8<5||4*3`	(A) -8	(B) 1	(C) 3	(D) 18	(E) 31	(F) 35	(G) 38	(H) 58
26/1p.	`5/7<4>=2+8`	(A) -90	(B) 0	(C) 1	(D) 5	(E) 13	(F) 61	(G) 74	(H) 79
27/1p.	`8+0-7/2+2`	(A) -51	(B) -33	(C) -3	(D) 0	(E) 2	(F) 3	(G) 7	(H) 73
28/1p.	`5*4/7%5*7`	(A) -47	(B) -6	(C) 0	(D) 1	(E) 2	(F) 14	(G) 35	(H) 77
29/1p.	`1-4*9-1%2`	(A) -44	(B) -34	(C) -31	(D) -28	(E) -24	(F) 0	(G) 4	(H) 30
30/1p.	`4+9*1>=3==2`	(A) -19	(B) -2	(C) 0	(D) 1	(E) 4	(F) 5	(G) 13	(H) 26
31/1p.	`3*5*8/4+1`	(A) -97	(B) -32	(C) 15	(D) 24	(E) 31	(F) 33	(G) 34	(H) 69

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

32/2p. `int a=-8; do body; while( a-- > -8 );`

33/2p. `int n; for( n=-3; n!=-2; ++n ) body;`

34/2p. `int u; for( u=0; u!=2; ++u ) body;`

35/2p. `int s=-1; do body; while( --s >= -2 );`

36/2p. `int t=10; while( --t <= 17 ) body;`

37/2p. `int b=6; do body; while( b-- < 10 );`

38/2p. `int p=2; do body; while( p++ < 8 );`

39/2p. `int y; for( y=2; y!=4; ++y ) 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) $n\sqrt{n}\lg n$ (E) $n\lg n$ (G) $\sqrt{n}\lg n$ (I) $\lg^2 n$
(B) n^2 (D) $n\sqrt{n}$ (F) n (H) $\sqrt{n}$ (J) $\lg n$

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      c = 1; do {
         simpleStatement;
      c += 2; } while ( c * c < 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) $n\lg n$ (G) $\sqrt{n}\lg n$ (I) $\lg n$
(B) n^2 (D) $n\sqrt{n}$ (F) n (H) $\lg^2 n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      b = n; do {
         g = n; do {
            simpleStatement;
         g--; } while ( g > 1 );
      b -= 5; } while ( b > 1 );
   } 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) $n^2\sqrt{n}$ (C) n^2 (E) $n\lg n$ (G) $\sqrt{n}$ (I) $\lg n$
(B) $n^2\lg n$ (D) $n\sqrt{n}\lg n$ (F) n (H) $\lg^2 n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      if ( simpleCompare ) {
         simpleStatement;
      } else {
         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^4 (C) $n^2\sqrt{n}$ (E) n^2 (G) $\sqrt{n}\lg^3 n$ (I) $\sqrt{n}$
(B) n^3 (D) $n^2 \lg n$ (F) $n\lg^3 n$ (H) $\sqrt{n}\lg n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   f = n; do {
      j = 1; while ( j * j < n ) {
         e = 1; while ( e * e < n ) {
            if ( simpleCompare ) {
               for ( d = 1 ; d < n ; d++ ) {
                  if ( simpleCompare ) {
                     simpleStatement;
                  } else {
                     simpleStatement;
                  }
               }
            } else {
               simpleStatement;
            }
         e += 2; }
      j++; }
   f--; } while ( f > 1 );
   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^4 (C) $n^2\sqrt{n}$ (E) $n\lg n$ (G) $\sqrt{n}\lg^3 n$ (I) $\lg^2 n$
(B) $n^3\sqrt{n}$ (D) $n\sqrt{n}\lg n$ (F) n (H) $\sqrt{n}$ (J) $\lg n$

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   c = 1; do {
      d = 1; do {
         k = 1; while ( k * k < n ) {
            if ( simpleCompare ) {
               if ( simpleCompare ) {
                  a = 1; do {
                     simpleStatement;
                  a += 3; } while ( a * a < n );
               }
            } else {
               simpleStatement;
            }
         k++; }
      d++; } while ( d * d < n );
   c *= 2; } while ( c < n );
   return 0; }
```

45/10p. 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) $n^4\sqrt{n}\lg^2 n$ (C) n^4 (E) $n^3\sqrt{n}$ (G) $n^2\sqrt{n}\lg^4 n$ (I) $n^2\lg^3 n$
(B) $n^4\lg n$ (D) $n^3\sqrt{n}\lg^2 n$ (F) $n^3\lg^4 n$ (H) $n^2\sqrt{n}$ (J) $n\lg^5 n$

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

Section II: List regex answers on a single well-labeled sheet of paper.

46/4p. Let L be the language accepted by the regular expression "`bb(aabbab)+`". List the shortest five (or all) strings in L.

47/4p. Let L be the language accepted by the regular expression "`bba(b?|aab)?`". List the shortest five (or all) strings in L.

48/4p. Let L be the language accepted by the regular expression "`bba(abbab)*`". List the shortest five (or all) strings in L.

49/4p. Let L be the language accepted by the regular expression "`(b(aab|a*)+baabb*baaba(a|b*)+)*`". List the shortest five (or all) strings in L.

Section III: Write each program on a separate well-labeled sheet of paper.

50/8p. Write this program on a separate sheet of paper. Use one side of one sheet. Label it clearly. Prompt for and accept two integers, a and b. Print their sum (add).

51/8p. Write this program on a separate sheet of paper. Use one side of one sheet. Label it clearly. Accept two integers, a and b, from the command line (argv). Print their sum (add).

52/15p. Write this program on a separate sheet of paper. Use one side of one sheet. Label it clearly. Print the smallest (min) number from the command line (argv).

53/15p. Write this program on a separate sheet of paper. Use one side of one sheet. Label it clearly. Prompt for and accept a dollar amount. Tell how to pay that amount using the smallest number of the following bills and coins: twenty, five, one, quarter, dime, nickel, penny. Example input: 36.19, output: twenty=1, five=3, one=1, dime=1, nickel=1, penny=4.

Total points 150.

Answer Key (points per line)

1	(2).	G
2	(2).	B
3	(2).	E
4	(2).	D
5	(2).	H
6	(2).	F
7	(2).	A
8	(2).	G
9	(2).	A
10	(2).	G
11	(2).	C
12	(2).	E
13	(1).	J (1)
14	(1).	G (1)
15	(1).	F (3)
16	(1).	E (1)
17	(1).	B (-6)
18	(1).	E (0)
19	(1).	C (-5)
20	(1).	E (2)
21	(1).	F (6)
22	(1).	E (14)
23	(1).	G (56)
24	(1).	G (4)
25	(1).	B (1)
26	(1).	B (0)
27	(1).	G (7)
28	(1).	F (14)
29	(1).	J (-36)
30	(1).	C (0)
31	(1).	E (31)
32	(2).	1
33	(2).	1
34	(2).	2
35	(2).	2
36	(2).	9
37	(2).	9
38	(2).	7
39	(2).	2
40	(3).	H ($\sqrt{n}$)
41	(3).	B (n^2)
42	(3).	J (1)
43	(5).	B (n^3)
44	(5).	D ($n\sqrt{n}\lg n$)
45	(10).	G ($n^2\sqrt{n}\lg^4 n$)

Total points 150.