Do NOT write on this test. Record all answers on the bubble sheet. **Closed book. No notes.** Work strictly from memory. **No calculators. No time limit. Scratch paper okay.**

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

1/1p.	`2-0%7-3*2`	(A) -4	(B) -2	(C) 0	(D) 2	(E) 4	(F) 10	(G) 26	(H) 71
2/1p.	`0-5+2%2%7`	(A) -82	(B) -28	(C) -5	(D) -1	(E) 1	(F) 2	(G) 6	(H) 37
3/1p.	`4%2*0/8+5`	(A) -6	(B) -4	(C) 0	(D) 5	(E) 17	(F) 21	(G) 42	(H) 75
4/1p.	`5+5*9/9/8`	(A) -55	(B) -42	(C) -41	(D) -17	(E) 0	(F) 50	(G) 90	(H) 94
5/1p.	`5/1*3/8-5`	(A) -80	(B) -25	(C) -5	(D) -4	(E) -1	(F) 5	(G) 7	(H) 77
6/1p.	`2+0-5-7+4`	(A) -74	(B) -61	(C) -14	(D) -6	(E) 0	(F) 7	(G) 8	(H) 32
7/1p.	`7+4%5*0-2`	(A) -65	(B) -22	(C) -9	(D) -1	(E) 5	(F) 7	(G) 59	(H) 86
8/1p.	`6-5%1%2*8`	(A) -84	(B) -74	(C) -44	(D) -31	(E) 1	(F) 6	(G) 48	(H) 61
9/1p.	`8/3-6*7/6`	(A) -92	(B) -48	(C) -7	(D) -5	(E) -2	(F) -1	(G) 67	(H) 95
10/1p.	`0-6-6+1-7`	(A) -73	(B) -18	(C) -8	(D) -6	(E) 6	(F) 8	(G) 84	(H) 88
11/1p.	`2+6/6/2-8`	(A) -8	(B) -7	(C) -6	(D) -4	(E) -2	(F) -1	(G) 42	(H) 57
12/1p.	`5-3<4<=7*7`	(A) -2	(B) 0	(C) 5	(D) 7	(E) 28	(F) 35	(G) 42	(H) 57
13/1p.	`9*2*5-7+2`	(A) -72	(B) -49	(C) -18	(D) 9	(E) 29	(F) 45	(G) 81	(H) 85
14/1p.	`5-4/4-0-5`	(A) -65	(B) -5	(C) -1	(D) 0	(E) 5	(F) 9	(G) 11	(H) 37
15/1p.	`6*5-6%8%8`	(A) -90	(B) -85	(C) -6	(D) 0	(E) 2	(F) 24	(G) 58	(H) 61
16/1p.	`5-2%2-8-3`	(A) -10	(B) -6	(C) 0	(D) 10	(E) 12	(F) 16	(G) 39	(H) 85
17/1p.	`6+0==0||3*7`	(A) -31	(B) -26	(C) 0	(D) 1	(E) 6	(F) 7	(G) 13	(H) 49
18/1p.	`5/5+5%3*5`	(A) -94	(B) -72	(C) -23	(D) 0	(E) 1	(F) 15	(G) 25	(H) 92
19/1p.	`4-1%8/4*7`	(A) -40	(B) -12	(C) -3	(D) 0	(E) 3	(F) 7	(G) 21	(H) 28
20/1p.	`5+3-6*1-7`	(A) -12	(B) -5	(C) 9	(D) 10	(E) 23	(F) 44	(G) 60	(H) 76
21/1p.	`3+6*0%6-3`	(A) -27	(B) -22	(C) -3	(D) 0	(E) 1	(F) 3	(G) 6	(H) 22
22/1p.	`4==8>9-2+9`	(A) -28	(B) -11	(C) 0	(D) 1	(E) 7	(F) 9	(G) 21	(H) 66
23/1p.	`9*0!=3||7-6`	(A) -67	(B) -45	(C) -36	(D) -5	(E) 1	(F) 2	(G) 3	(H) 56
24/1p.	`0*7-5+5+7`	(A) -94	(B) -37	(C) -17	(D) -3	(E) 7	(F) 12	(G) 61	(H) 97
25/1p.	`0-0<=7&&9-9`	(A) -20	(B) -10	(C) -1	(D) 0	(E) 1	(F) 8	(G) 43	(H) 54
26/1p.	`9-3+0*6/8`	(A) -26	(B) -23	(C) -2	(D) 0	(E) 4	(F) 7	(G) 9	(H) 56
27/1p.	`3-6!=8>=7-5`	(A) -5	(B) -4	(C) -2	(D) 1	(E) 2	(F) 3	(G) 7	(H) 8
28/1p.	`4+7-6/5+1`	(A) -10	(B) -5	(C) 0	(D) 2	(E) 4	(F) 5	(G) 10	(H) 11
29/1p.	`5*3/4%5+3`	(A) -72	(B) -51	(C) -46	(D) 0	(E) 3	(F) 6	(G) 19	(H) 66
30/1p.	`6+2==4>=6-9`	(A) -56	(B) -21	(C) -8	(D) -3	(E) 0	(F) 1	(G) 7	(H) 48
31/1p.	`3+9+2*6%2`	(A) -94	(B) -53	(C) -7	(D) 3	(E) 4	(F) 12	(G) 66	(H) 93
32/1p.	`4%1/1-9-8`	(A) -88	(B) -61	(C) -29	(D) -17	(E) -12	(F) -8	(G) -1	(H) 47
33/1p.	`3%9%7%6+9`	(A) -94	(B) -8	(C) 1	(D) 9	(E) 12	(F) 78	(G) 79	(H) 90
34/1p.	`1-8+1*2-0`	(A) -55	(B) -17	(C) -13	(D) -12	(E) -9	(F) -5	(G) 30	(H) 70

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

(A) `$` (B) `*` (C) `+` (D) `<` (E) `\0dd` (F) `\e` (G) `\xdd` (H) `esc` (I) `{`

35/1p. octal dd
36/1p. end of string
37/1p. the escape character
38/1p. repeat one or more times
39/1p. start of multiplier

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

(A) `$` (B) `+/-` (C) `?` (D) `\*` (E) `\D` (F) `\d` (G) `^` (H) `{n,m}` (I) `{n..m}`

40/1p. the null character
41/1p. non-digit
42/1p. at least n times, at most m
43/1p. start of string
44/1p. repeat zero or one times

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

45/1p. Does the empty string match the regular expression "`ps`"?

46/1p. Does the string "`bc`" match the regular expression "`hz|bc`"?

47/1p. Does the string "`pnrn`" match the regular expression "`pn+rn`"?

48/1p. Does the string "`zp`" match the regular expression "`zp|kr`"?

49/1p. Does the empty string match the regular expression "`hz*c*`"?

50/1p. Does the string "`tta`" match the regular expression "`(t+|a)?`"?

51/1p. Does the string "`kzkzn`" match the regular expression "`(kz+|n)+`"?

52/1p. Does the string "`fnn`" match the regular expression "`(ud+|fn)*`"?

53/1p. Does the string "`bdcfk`" match the regular expression "`b?dc*fk`"?

54/1p. Does the string "`dxdxgccgcc`" match the regular expression "`dx+|((gc)+)+`"?

55/1p. Does the string "`wwrb`" match the regular expression "`w+|rb|rz`"?

56/1p. Does the string "`t`" match the regular expression "`(tf)+`"?

57/1p. Does the string "`wwn`" match the regular expression "`(w*g)*n?`"?

58/1p. Does the string "`hp`" match the regular expression "`(pc)*|(h+pp)*`"?

59/1p. Does the string "`faubz`" match the regular expression "`fa?ub|z|ht`"?

60/1p. Does the string "`bqh`" match the regular expression "`(bq*|ht)?`"?

61/1p. Does the string "`hs`" match the regular expression "`(h+|s|ff)?|h*`"?

62/1p. Does the string "`xkukkknbkbk`" match the regular expression "`(xk)+(uk)*kn(bk)?`"?

63/1p. Does the empty string match the regular expression "`(rz+|k?|(qh)?)?`"?

64/1p. Does the string "`hn`" match the regular expression "`t?hn|(gb*pf)*`"?

65/1p. Does the string "`ktxssp`" match the regular expression "`(kt|xs)+(pq)?|g+`"?

66/1p. Does the string "`dputqnqn`" match the regular expression "`(dp)*(ut)*|((gq)+|qn)*`"?

67/1p. Does the string "`wawawrrd`" match the regular expression "`(wa)+wr*(rd|w)?`"?

68/1p. Does the string "`fdcsb`" match the regular expression "`(bn)*|hh|fd|(cf)+(sb)*`"?

69/2p. 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) $\sqrt{n} \lg n$ (I) $\lg n$
(B) $n^2 \lg n$ (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 ) {
      d = 1; while ( d * d < n ) {
         f = 1; while ( f * f < n ) {
            simpleStatement;
         f++; }
      d++; }
   } else {
      simpleStatement;
   }
   return 0; }
```

70/2p. 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) $\sqrt{n} \lg n$ (H) $\lg^2 n$ (J) 1

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

71/2p. 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) $n^2 \lg n$ (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) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   f = 1; while ( f < n ) {
      if ( simpleCompare ) {
         for ( a = 1 ; a * a < n ; a += 5 ) {
            if ( simpleCompare ) {
               simpleStatement;
            }
         }
      } else {
         simpleStatement;
      }
   f *= 2; }
   return 0; }
```

72/2p. 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) $n^3 \lg n$ (C) $n^2 \lg^3 n$ (E) $n^2 \lg n$ (G) $n \lg n$ (I) $\lg n$
(B) $n^2\sqrt{n} \lg n$ (D) $n^2 \lg^2 n$ (F) $n\sqrt{n} \lg^3 n$ (H) $\sqrt{n} \lg n$ (J) 1

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

73/2p. 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) $n^2 \lg n$ (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]);
   f = 1; do {
      if ( simpleCompare ) {
         if ( simpleCompare ) {
            simpleStatement;
         } else {
            simpleStatement;
         }
      }
   f++; } while ( f * f < n );
   return 0; }
```

74/2p. 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^3 (C) n^2 (E) $n\sqrt{n}$ (G) $n \lg n$ (I) $\sqrt{n}$
(B) $n^2\sqrt{n}$ (D) $n\sqrt{n} \lg n$ (F) $n \lg^2 n$ (H) $\sqrt{n} \lg n$ (J) 1

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

75/2p. 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 ) {
         i = n; while ( i > 1 ) {
            if ( simpleCompare ) {
               if ( simpleCompare ) {
                  simpleStatement;
               }
            } else {
               simpleStatement;
            }
         i /= 2; }
      }
   }
   return 0; }
```

76/2p. 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) $\sqrt{n} \lg n$ (I) $\lg n$
(B) $n^2 \lg n$ (D) $n\sqrt{n} \lg n$ (F) $n \lg n$ (H) $\lg^2 n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      if ( simpleCompare ) {
         c = n; while ( c > 1 ) {
            e = n; do {
               simpleStatement;
            e -= 5; } while ( e > 1 );
         c /= 2; }
      } else {
         simpleStatement;
      }
   } else {
      if ( simpleCompare ) {
         g = n; do {
            simpleStatement;
         g /= 2; } while ( g > 1 );
      } else {
         simpleStatement;
      }
   }
   return 0; }
```

77/2p. 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) $n^2\sqrt{n}\lg n$ (E) $n^2\lg n$ (G) $n\sqrt{n}\lg n$ (I) $\sqrt{n}\lg n$
(B) $n^3\sqrt{n}$ (D) $n^2\sqrt{n}$ (F) n^2 (H) $n\lg^3 n$ (J) $\sqrt{n}$

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

78/2p. 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) $n^3\lg n$ (C) $n^2\lg^2 n$ (E) n^2 (G) $n\sqrt{n}\lg n$ (I) $\lg^4 n$
(B) $n^2\sqrt{n}\lg n$ (D) $n^2\lg n$ (F) $n\sqrt{n}\lg^2 n$ (H) $n\lg n$ (J) $\lg n$

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   d = 1; while ( d < n ) {
      e = 1; do {
         f = n; do {
            if ( simpleCompare ) {
               if ( simpleCompare ) {
                  if ( simpleCompare ) {
                     simpleStatement;
                  } else {
                     simpleStatement;
                  }
               }
            } else {
               c = 1; do {
                  simpleStatement;
               c *= 5; } while ( c < n );
            }
         f -= 5; } while ( f > 1 );
      e += 3; } while ( e * e < n );
   d *= 3; }
   return 0; }
```

79/2p. 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) $\sqrt{n}$ (J) 1

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

80/2p. 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^3 (C) $n^2\lg n$ (E) $n\sqrt{n}$ (G) $n\lg n$ (I) $\lg^3 n$
(B) $n^2\sqrt{n}$ (D) $n\sqrt{n}\lg n$ (F) $n\lg^2 n$ (H) $\sqrt{n}$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   for ( i = 1 ; i * i < n ; i += 1 ) {
      g = 1; while ( g < n ) {
         if ( simpleCompare ) {
            if ( simpleCompare ) {
               e = n; do {
                  simpleStatement;
               e -= 5; } while ( e > 1 );
            } else {
               simpleStatement;
            }
         } else {
            if ( simpleCompare ) {
               simpleStatement;
            } else {
               simpleStatement;
            }
         }
      g++; }
   }
   return 0; }
```

81/2p. 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) $n^3 \lg n$ (E) $n^2 \lg n$ (G) $n\sqrt{n}$ (I) $n \lg n$
(B) $n^3\sqrt{n}$ (D) $n^2\sqrt{n}$ (F) n^2 (H) $n \lg^3 n$ (J) $\lg^2 n$

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   for ( g = 1 ; g < n ; g++ ) {
      if ( simpleCompare ) {
         if ( simpleCompare ) {
            if ( simpleCompare ) {
               for ( j = 1 ; j < n ; j += 1 ) {
                  if ( simpleCompare ) {
                     simpleStatement;
                  } else {
                     simpleStatement;
                  }
               }
            }
         } else {
            if ( simpleCompare ) {
               a = 1; do {
                  simpleStatement;
               a++; } while ( a * a < n );
            } else {
               simpleStatement;
            }
         }
      } else {
         b = 1; do {
            if ( simpleCompare ) {
               if ( simpleCompare ) {
                  e = 1; while ( e < n ) {
                     i = 1; do {
                        simpleStatement;
                     i++; } while ( i * i < n );
                  e *= 5; }
               } else {
                  simpleStatement;
               }
            }
         b += 5; } while ( b * b < n );
      }
   }
   return 0; }
```

82/2p. 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) $n^3 \lg n$ (C) $n^2\sqrt{n}\lg^2 n$ (E) $n^2 \lg n$ (G) $n \lg^4 n$ (I) $\sqrt{n}\lg^4 n$
(B) n^3 (D) $n^2\sqrt{n}$ (F) $n\sqrt{n}\lg n$ (H) $n \lg^3 n$ (J) $\lg^5 n$

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      for ( h = n ; h > 1 ; h -= 10 ) {
         k = 1; while ( k < n ) {
            if ( simpleCompare ) {
               if ( simpleCompare ) {
                  d = 1; do {
                     if ( simpleCompare ) {
                        for ( a = n ; a > 1 ; a /= 2 ) {
                           if ( simpleCompare ) {
                              simpleStatement;
                           }
                        }
                     } else {
                        if ( simpleCompare ) {
                           simpleStatement;
                        }
                     }
                  d++; } while ( d * d < n );
               }
            } else {
               i = n; while ( i > 1 ) {
                  if ( simpleCompare ) {
                     for ( g = 1 ; g * g < n ; g++ ) {
                        for ( b = 1 ; b * b < n ; b++ ) {
                           simpleStatement;
                        }
                     }
                  } else {
                     if ( simpleCompare ) {
                        simpleStatement;
                     }
                  }
               i /= 2; }
            }
         k++; }
      }
   }
   return 0; }
```

83/2p. 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) $n^3\sqrt{n}$ (C) $n^2\sqrt{n}\lg n$ (E) n^2 (G) $n\lg^3 n$ (I) n
(B) $n^3\lg n$ (D) $n^2\lg^2 n$ (F) $n\sqrt{n}\lg^2 n$ (H) $n\lg n$ (J) $\lg^4 n$

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

84/2p. 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) $\lg^2 n$
(B) $n^2 \lg n$ (D) $n\sqrt{n}\lg n$ (F) $n\lg^2 n$ (H) $\sqrt{n}\lg^2 n$ (J) 1

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

Total points 100.

Answer Key (points per line)

1 (1).	A (-4)	43 (1).	G
2 (1).	C (-5)	44 (1).	C
3 (1).	D (5)	45 (1).	false
4 (1).	J (5)	46 (1).	true
5 (1).	D (-4)	47 (1).	true
6 (1).	D (-6)	48 (1).	true
7 (1).	E (5)	49 (1).	false
8 (1).	F (6)	50 (1).	false
9 (1).	D (-5)	51 (1).	true
10 (1).	B (-18)	52 (1).	false
11 (1).	C (-6)	53 (1).	true
12 (1).	J (1)	54 (1).	false
13 (1).	H (85)	55 (1).	false
14 (1).	C (-1)	56 (1).	false
15 (1).	F (24)	57 (1).	false
16 (1).	B (-6)	58 (1).	false
17 (1).	D (1)	59 (1).	false
18 (1).	J (11)	60 (1).	false
19 (1).	J (4)	61 (1).	false
20 (1).	B (-5)	62 (1).	false
21 (1).	D (0)	63 (1).	true
22 (1).	C (0)	64 (1).	true
23 (1).	E (1)	65 (1).	false
24 (1).	E (7)	66 (1).	false
25 (1).	D (0)	67 (1).	true
26 (1).	J (6)	68 (1).	false
27 (1).	D (1)	69 (2).	F (n)
28 (1).	H (11)	70 (2).	G ($\sqrt{n}$)
29 (1).	F (6)	71 (2).	G ($\sqrt{n}\lg n$)
30 (1).	E (0)	72 (2).	A ($n^3 \lg n$)
31 (1).	F (12)	73 (2).	G ($\sqrt{n}$)
32 (1).	D (-17)	74 (2).	D ($n\sqrt{n}\lg n$)
33 (1).	E (12)	75 (2).	I ($\lg n$)
34 (1).	F (-5)	76 (2).	F ($n \lg n$)
35 (1).	E	77 (2).	E ($n^2 \lg n$)
36 (1).	A	78 (2).	F ($n\sqrt{n}\lg^2 n$)
37 (1).	F	79 (2).	J (1)
38 (1).	C	80 (2).	B ($n^2\sqrt{n}$)
39 (1).	I	81 (2).	E ($n^2 \lg n$)
40 (1).	J	82 (2).	A ($n^3 \lg n$)
41 (1).	E	83 (2).	F ($n\sqrt{n}\lg^2 n$)
42 (1).	H	84 (2).	D ($n\sqrt{n}\lg n$)

Total points 100.