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.	`3/2%1-3*3`	(A) -35	(B) -20	(C) -9	(D) -7	(E) -3	(F) -1	(G) 20	(H) 53
2/1p.	`1/7*0/8-9`	(A) -66	(B) -46	(C) -1	(D) 0	(E) 26	(F) 49	(G) 66	(H) 82
3/1p.	`0/7%4+3-1`	(A) -28	(B) -21	(C) -15	(D) -1	(E) 0	(F) 2	(G) 36	(H) 57
4/1p.	`1+6-1/7+3`	(A) -85	(B) 0	(C) 3	(D) 4	(E) 7	(F) 10	(G) 72	(H) 91
5/1p.	`0-1+6/5-0`	(A) -85	(B) -84	(C) -43	(D) -2	(E) -1	(F) 1	(G) 70	(H) 72
6/1p.	`9-9%2+7*4`	(A) -50	(B) -23	(C) -20	(D) 9	(E) 28	(F) 36	(G) 60	(H) 90
7/1p.	`5/3-8*7/3`	(A) -60	(B) -37	(C) -19	(D) -17	(E) -14	(F) -3	(G) -2	(H) -1
8/1p.	`1/2-3*4%3`	(A) -60	(B) -3	(C) -1	(D) 0	(E) 2	(F) 65	(G) 84	(H) 98
9/1p.	`3/4/6+5+2`	(A) -79	(B) -66	(C) -21	(D) -13	(E) 0	(F) 1	(G) 7	(H) 71
10/1p.	`7+0-6*6-9`	(A) -41	(B) -38	(C) -21	(D) -20	(E) -17	(F) -8	(G) -3	(H) 25
11/1p.	`2+1<6>=5*5`	(A) -74	(B) -42	(C) -23	(D) 0	(E) 1	(F) 2	(G) 3	(H) 7
12/1p.	`2/9-8+5%9`	(A) -13	(B) -3	(C) -1	(D) 0	(E) 5	(F) 6	(G) 8	(H) 47
13/1p.	`2%4-5/2+4`	(A) -72	(B) -44	(C) -4	(D) -1	(E) 0	(F) 4	(G) 96	(H) 99
14/1p.	`9-4%5%8+2`	(A) -95	(B) 2	(C) 3	(D) 5	(E) 7	(F) 34	(G) 67	(H) 89
15/1p.	`9+7/7%9/2`	(A) -97	(B) 0	(C) 1	(D) 2	(E) 9	(F) 10	(G) 22	(H) 83
16/1p.	`4%7-3%8+8`	(A) -54	(B) -23	(C) -7	(D) 0	(E) 1	(F) 4	(G) 9	(H) 51
17/1p.	`4-1-8*0+4`	(A) -29	(B) -20	(C) -1	(D) 0	(E) 4	(F) 7	(G) 8	(H) 35
18/1p.	`5-0>7<7-8`	(A) -72	(B) -7	(C) -4	(D) 0	(E) 1	(F) 5	(G) 12	(H) 13
19/1p.	`8-1>=1<8-8`	(A) -55	(B) -7	(C) 0	(D) 1	(E) 7	(F) 8	(G) 15	(H) 91
20/1p.	`2/9&&0>2-9`	(A) -46	(B) -9	(C) -1	(D) 0	(E) 1	(F) 31	(G) 40	(H) 52
21/1p.	`5-7>0!=5+5`	(A) -56	(B) -23	(C) 0	(D) 1	(E) 4	(F) 5	(G) 6	(H) 9
22/1p.	`5%4%7*3+8`	(A) -24	(B) -13	(C) -4	(D) 1	(E) 9	(F) 11	(G) 15	(H) 93
23/1p.	`6%9+5*4%6`	(A) 0	(B) 2	(C) 8	(D) 24	(E) 26	(F) 44	(G) 63	(H) 76
24/1p.	`1-3%0-1+2`	(A) -32	(B) -26	(C) -4	(D) -1	(E) 0	(F) 1	(G) 2	(H) 3
25/1p.	`8+9==7&&0+3`	(A) -96	(B) -86	(C) -69	(D) 0	(E) 1	(F) 3	(G) 4	(H) 84
26/1p.	`2%8*8+5+2`	(A) -11	(B) 2	(C) 4	(D) 9	(E) 23	(F) 28	(G) 92	(H) 97
27/1p.	`9+3/7/3*2`	(A) 0	(B) 1	(C) 6	(D) 9	(E) 12	(F) 18	(G) 20	(H) 93
28/1p.	`9-1-2-0%1`	(A) -83	(B) -35	(C) 6	(D) 8	(E) 9	(F) 10	(G) 52	(H) 60
29/1p.	`6*6-5%2-1`	(A) -76	(B) -1	(C) 0	(D) 5	(E) 16	(F) 24	(G) 34	(H) 36
30/1p.	`8%2||4<=0-4`	(A) -40	(B) -4	(C) -3	(D) -1	(E) 1	(F) 16	(G) 49	(H) 53
31/1p.	`9/1%7%6*4`	(A) -88	(B) -48	(C) 0	(D) 1	(E) 8	(F) 12	(G) 36	(H) 46
32/1p.	`1-9/6+2*3`	(A) -32	(B) -8	(C) -6	(D) -2	(E) -1	(F) 0	(G) 1	(H) 4
33/1p.	`9&&3!=3-5/2`	(A) -85	(B) -27	(C) -18	(D) -3	(E) -1	(F) 0	(G) 24	(H) 75
34/1p.	`8+5-9==0>3`	(A) -91	(B) -9	(C) 0	(D) 1	(E) 8	(F) 9	(G) 22	(H) 73

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

(A) `<` (B) `\D` (C) `\W` (D) `\d` (E) `\n` (F) `\r` (G) `\w` (H) `{` (I) `|`

35/1p. digit
36/1p. newline
37/1p. not a word character
38/1p. start of multiplier
39/1p. or (alternation)

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

(A) `)` (B) `[` (C) `\0dd` (D) `\2` (E) `\D` (F) `\d` (G) `\s` (H) `\xdd` (I) `eog`

40/1p. octal dd
41/1p. second back-reference
42/1p. non-digit
43/1p. start of character class
44/1p. end of group

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

45/1p. Does the string "`rn`" match the regular expression "`r+n`"?

46/1p. Does the string "`xbb`" match the regular expression "`xb+|hf`"?

47/1p. Does the string "`yyywd`" match the regular expression "`yy*wd`"?

48/1p. Does the string "`baaa`" match the regular expression "`(ba)+a`"?

49/1p. Does the string "`dks`" match the regular expression "`(d+ks)?`"?

50/1p. Does the string "`sd`" match the regular expression "`zr+|sd`"?

51/1p. Does the string "`kttk`" match the regular expression "`(kt)?k|qn`"?

52/1p. Does the string "`xu`" match the regular expression "`(xu)+(ss)+`"?

53/1p. Does the string "`tktkakzkz`" match the regular expression "`(tk)*|a|kz?`"?

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

55/1p. Does the empty string match the regular expression "`t(hf*)*`"?

56/1p. Does the string "`k`" match the regular expression "`k*|xt?b`"?

57/1p. Does the empty string match the regular expression "`(x*tc*)+`"?

58/1p. Does the string "`hthty`" match the regular expression "`(hp)+|fc|ht+|y`"?

59/1p. Does the string "`ryhryh`" match the regular expression "`x?|(ry|h)*`"?

60/1p. Does the string "`g`" match the regular expression "`(xg)*|bh|(g*)?`"?

61/1p. Does the string "`zbbppppxxwk`" match the regular expression "`(zb)?|pp+x*wk`"?

62/1p. Does the string "`nwgtfx`" match the regular expression "`n+|wg|t+|fx`"?

63/1p. Does the string "`dppbnbn`" match the regular expression "`dp*(bn)*|y`"?

64/1p. Does the string "`prrnnrq`" match the regular expression "`((pr)*|rn)*nr*q`"?

65/1p. Does the string "`rqrrgrgg`" match the regular expression "`((rq)+r|(rg)?|(aq)*)*`"?

66/1p. Does the string "`ysysyd`" match the regular expression "`qf+|ys+yd`"?

67/1p. Does the string "`wxwxkgw`" match the regular expression "`t?(wx)*|(k*gw)?`"?

68/1p. Does the string "`zhhpxhkzhhpxhk`" match the regular expression "`((zh)?px|(gd)*|hk)*`"?

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) $\lg^2 n$
(B) $n^2 \lg n$ (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]);
   if ( simpleCompare ) {
      if ( simpleCompare ) {
         for ( i = n ; i > 1 ; i /= 3 ) {
            simpleStatement;
         }
      } else {
         simpleStatement;
      }
   } 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) $n\sqrt{n}$ (G) $\sqrt{n} \lg n$ (I) $\lg n$
(B) $n^2 \lg n$ (D) $n\sqrt{n} \lg n$ (F) n (H) $\sqrt{n}$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      if ( simpleCompare ) {
         for ( i = 1 ; i < n ; i++ ) {
            if ( simpleCompare ) {
               simpleStatement;
            }
         }
      } else {
         for ( k = 1 ; k * k < n ; k += 5 ) {
            simpleStatement;
         }
      }
   } else {
      j = 1; do {
         d = 1; do {
            simpleStatement;
         d++; } while ( d * d < n );
      j *= 3; } while ( j < n );
   }
   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) $\lg n$
(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 ) {
      for ( c = 1 ; c * c < n ; c++ ) {
         for ( i = 1 ; i < n ; i *= 3 ) {
            simpleStatement;
         }
      }
   } else {
      for ( e = 1 ; e < n ; e += 3 ) {
         simpleStatement;
      }
   }
   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) $n^2 \lg n$ (C) $n\sqrt{n}$ (E) n (G) $\sqrt{n}$ (I) $\lg n$
(B) n^2 (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]);
   if ( simpleCompare ) {
      for ( i = 1 ; i < n ; i += 3 ) {
         if ( simpleCompare ) {
            if ( simpleCompare ) {
               if ( simpleCompare ) {
                  simpleStatement;
               } else {
                  simpleStatement;
               }
            }
         } else {
            simpleStatement;
         }
      }
   } else {
      h = 1; do {
         j = 1; while ( j * j < n ) {
            simpleStatement;
         j += 1; }
      h += 5; } while ( h * 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) $n^2\sqrt{n}$ (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 ) {
      if ( simpleCompare ) {
         c = n; while ( c > 1 ) {
            if ( simpleCompare ) {
               simpleStatement;
            }
         c--; }
      } else {
         simpleStatement;
      }
   } else {
      i = n; while ( i > 1 ) {
         simpleStatement;
      i /= 3; }
   }
   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) $n^2\sqrt{n}$ (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 ) {
      if ( simpleCompare ) {
         if ( simpleCompare ) {
            for ( k = n ; k > 1 ; k /= 3 ) {
               simpleStatement;
            }
         } else {
            simpleStatement;
         }
      }
   } else {
      for ( j = n ; j > 1 ; j -= 10 ) {
         for ( i = 1 ; i * i < n ; i += 5 ) {
            simpleStatement;
         }
      }
   }
   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) $n^2 \lg n$ (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 ) {
      if ( simpleCompare ) {
         if ( simpleCompare ) {
            simpleStatement;
         }
      } else {
         simpleStatement;
      }
   } else {
      simpleStatement;
   }
   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^3 (C) $n^2 \lg n$ (E) $n\sqrt{n} \lg n$ (G) $n \lg^2 n$ (I) $\sqrt{n} \lg^2 n$
(B) $n^2\sqrt{n}$ (D) n^2 (F) $n\sqrt{n}$ (H) n (J) $\sqrt{n}$

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

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   g = 1; while ( g < n ) {
      j = 1; while ( j * j < n ) {
         if ( simpleCompare ) {
            a = 1; do {
               for ( i = 1 ; i < n ; i++ ) {
                  simpleStatement;
               }
            a++; } while ( a < n );
         }
      j += 2; }
   g += 10; }
   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) $n^2\sqrt{n}$ (C) $n\sqrt{n} \lg n$ (E) $n \lg^2 n$ (G) n (I) $\sqrt{n} \lg n$
(B) $n^2 \lg n$ (D) $n\sqrt{n}$ (F) $n \lg n$ (H) $\sqrt{n} \lg^2 n$ (J) $\lg^2 n$

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   for ( b = 1 ; b < n ; b *= 2 ) {
      if ( simpleCompare ) {
         if ( simpleCompare ) {
            if ( simpleCompare ) {
               for ( i = n ; i > 1 ; i -= 5 ) {
                  j = 1; while ( j * j < n ) {
                     simpleStatement;
                  j++; }
               }
            } else {
               simpleStatement;
            }
         } else {
            if ( simpleCompare ) {
               simpleStatement;
            } else {
               simpleStatement;
            }
         }
      }
   }
   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) $n\sqrt{n}$ (E) n (G) $\sqrt{n}$ (I) $\lg n$
(B) n^2 (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]);
   if ( simpleCompare ) {
      e = n; while ( e > 1 ) {
         g = 1; do {
            simpleStatement;
         g++; } while ( g < n );
      e /= 2; }
   } else {
      if ( simpleCompare ) {
         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) $n^2\sqrt{n}$ (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) $\lg^2 n$

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

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

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   f = n; while ( f > 1 ) {
      if ( simpleCompare ) {
         j = 1; while ( j * j < n ) {
            i = 1; do {
               if ( simpleCompare ) {
                  e = 1; do {
                     if ( simpleCompare ) {
                        simpleStatement;
                     } else {
                        simpleStatement;
                     }
                  e++; } while ( e * e < n );
               } else {
                  if ( simpleCompare ) {
                     simpleStatement;
                  } else {
                     simpleStatement;
                  }
               }
            i++; } while ( i < n );
         j += 10; }
      } else {
         for ( d = 1 ; d * d < n ; d++ ) {
            if ( simpleCompare ) {
               if ( simpleCompare ) {
                  if ( simpleCompare ) {
                     for ( c = 1 ; c * c < n ; c++ ) {
                        simpleStatement;
                     }
                  } else {
                     simpleStatement;
                  }
               }
            }
         }
      }
   f -= 3; }
   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 (C) $n^2 \lg n$ (E) $n\sqrt{n} \lg n$ (G) $n \lg^2 n$ (I) n
(B) $n^2\sqrt{n}$ (D) n^2 (F) $n\sqrt{n}$ (H) $n \lg n$ (J) $\sqrt{n} \lg^2 n$

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      if ( simpleCompare ) {
         h = 1; while ( h * h < n ) {
            if ( simpleCompare ) {
               b = 1; do {
                  if ( simpleCompare ) {
                     if ( simpleCompare ) {
                        simpleStatement;
                     }
                  } else {
                     simpleStatement;
                  }
               b++; } while ( b * b < n );
            } else {
               if ( simpleCompare ) {
                  f = 1; while ( f < n ) {
                     simpleStatement;
                  f *= 2; }
               } else {
                  simpleStatement;
               }
            }
         h++; }
      } else {
         g = 1; while ( g < n ) {
            c = 1; while ( c < n ) {
               a = 1; do {
                  if ( simpleCompare ) {
                     if ( simpleCompare ) {
                        simpleStatement;
                     } else {
                        simpleStatement;
                     }
                  }
               a++; } while ( a * a < n );
            c++; }
         g *= 3; }
      }
   }
   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) $n \lg n$ (I) $\lg^2 n$
(B) $n^2 \lg n$ (D) $n\sqrt{n} \lg n$ (F) $n \lg^2 n$ (H) $\sqrt{n} \lg n$ (J) 1

```
int main ( int argc, char * * argv ) {
   int n = atoi(argv[1]);
   if ( simpleCompare ) {
      a = 1; do {
         if ( simpleCompare ) {
            if ( simpleCompare ) {
               if ( simpleCompare ) {
                  if ( simpleCompare ) {
                     simpleStatement;
                  }
               }
            } else {
               f = n; while ( f > 1 ) {
                  simpleStatement;
               f--; }
            }
         } else {
            c = 1; do {
               k = 1; do {
                  simpleStatement;
               k += 2; } while ( k * k < n );
            c++; } while ( c * c < n );
         }
      a *= 5; } while ( a < n );
   } else {
      j = n; while ( j > 1 ) {
         if ( simpleCompare ) {
            g = 1; do {
               if ( simpleCompare ) {
                  for ( i = 1 ; i < n ; i *= 2 ) {
                     simpleStatement;
                  }
               } else {
                  simpleStatement;
               }
            g += 10; } while ( g * g < n );
         } else {
            d = 1; do {
               simpleStatement;
            d++; } while ( d * d < n );
         }
      j /= 2; }
   }
   return 0; }
```

Total points 100.

Answer Key (points per line)

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

Total points 100.