UNIVERSITY OF MANITOBA

DEPARTMENT OF COMPUTER SCIENCE

Manitoba High School Programming Contest 2018

May 25 2018 12:30 – 3:30

Contest Rules:

- Do not open this package until instructed to do so.
- All solutions must be entered completely during the contest. No electronic copies of pre-written code are permitted.
- You may submit as many solutions as you like to each problem; however, incorrect solutions will be assessed a time penalty.
- Contest score is based on the most problems successfully solved; ties are won by shortest total time taken, including any penalties.
- A correct submission must solve the given problem and produce correct output for the given test data within a reasonable time.
- Programming style will not be considered during judging.
- Any programming language resources and notes are allowed.
- No other Internet access is allowed during the contest.

Submission Requirements / Pre-submission Checklist:

A. All input must be read from standard input (in Java, `System.in`). Only Problem 1 does not require input. **Do not open input files.**
B. All output must be written to standard output (in Java, use `System.out.print` or `println`).
C. Your output format must follow the problem requirements **exactly**.
D. Submit the source code file (`.java`, `.c`, etc.–**NOT** `.class`, `.exe`, etc.).
E. Java programs must be in a single file and not placed in a package (**no package statements**; `import` statements are OK).
F. Java programs must be complete; **they require a `main` method**.

NOTE: TEXT IN RED WAS FIXED DURING THE COMPETITION.

Problem 1 – Positively Functional

You have done an experiment and modeled a certain phenomena by the function

$$f(t) = \frac{100\sin(t)}{t^3 - 30t^2 + 200t}$$

You are interested in those values of *t* that are whole numbers between 1 and 100 where this function is defined and the value of *f(t)* is greater than zero.

Write a program that prints out only the values of *t* where the value of the function is greater than zero.

For instance, at *t*=1, the value of the function is (approximately) 0.4921, but at *t*=4, the value of the function is -0.1971. Thus, your program should print out 1, but not 4. Similarly, your program should print out 97 since the value of the function at *t*=97 is 0.0000584.

Input

This program does not have any input.

Output

Write the values t between 1 and 100 (inclusive) where the function f(t) is defined and the value of f(t) is greater than zero. Output each value of t on its own line.

Sample Output
1 2 3 7 [some values deleted] 97

Problem 2 – Passwords

You work as a programmer for an e-commerce website. Your company requires all your customers to create an account with a username and password. Your boss, who is not a computer scientist, wants to make sure that the passwords that are used are "secure" by insisting on more and more elaborate requirements. Each week, he has a new requirement. The first few requirements were ok, like "the password must contain at least eight characters" and "the password must contain a letter and a number". But now things have gotten out of hand. This week, your boss has instituted a new rule: "every password must contain two upper case Ts anywhere in the password, or a lower case t followed at some point by a lower case o." Under this new rule, `TrT135a6` would be acceptable, as would `abct1abco`, but `arTsandCrafts` would not.

Write a program that accepts a single string and determines if it satisfies your boss' most recent rule. You don't need to worry about other rules, since you've already written programs to check all of them.

Input

The input is a single string of characters. The characters are letters (upper case and lower case), digits, and the special characters

```
, . / < > ? ; : ' " [ { ] } \ | ` ~ !
@ # $ % ^ & * ( ) - _ = +
```

There are no spaces, tabs or other whitespace in the characters.

Output

For the password, output either `Password 'good'` or `Password 'bad'`, including the single quotation marks, on a line.

Sample Input 1	Sample Output 1
`arTsandCrafts`	`Password 'bad'`

Sample Input 2	Sample Output 2
`oTTer`	`Password 'good'`

Judging Data for Problem 2

Case	Input
1	arTsandCrafts
2	oTTer
3	otter
4	TrT135a6
5	abct1abco
6	otT.??tt
7	WriTeApRoGRaMtHaTacCepTSaStrINg
8	__][__PASSWORD___]
9	Totally
10	ToTally

Problem 3 – Brainheck

Brainheck is an esoteric programming language that has only 8 commands, encoded as single characters: `<>+-.,[]`. Any other characters in a Brainheck program (including whitespace) are ignored.

In short, a program written in Brainheck has:

- Access to an infinitely large array,
- A pointer that indicates the current position in the array, with instructions to move left `<` or right `>` in the array,
- Instructions to add `+` or subtract `-` a value from the current position in the array,
- Instructions to print `.` or read into `,` the current position in the array,
- Instructions that resemble the behaviour of a *while loop*: `[` and `]`.

A program written in Brainheck is valid if, and only if, every `[` has a matching `]` that appears after (to the right) of it in the program. There are **no** other restrictions on the language.

Your job is to write a Brainheck validator. Your program should read in an entire Brainheck program and print `VALID` if the program is a valid Brainheck program or `INVALID` if the program is **not** a valid Brainheck program.

Input

The input is a Brainheck program on a single line of input. (In the sample inputs below, long lines are word-wrapped.)

Output

Output either `VALID` or `INVALID` as described above.

Sample Input 1	Sample Output 1
`[->+<]`	`VALID`

Sample Input 2	Sample Output 2
`][->][`	`INVALID`

Sample Input 3	Sample Output 3
`++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---.+++++++..+++.>>.<-.<.+++.------.--------.>>+.>++.`	`VALID`

Judging Data for Problem 3

Case	Input
1	`[->+<]`
2	`][->][`
3	`++++++++[>++++[>++>+++>+++>+<<<<-]>+>+>->>+[<]<-]>>.>---.+++++++..+++.>>.<-.<.+++.------.--------.>>+.>++.`
4	`-+-`
5	`+a-b-c+`
6	`>.<`
7	`[]`
8	`[][]`
9	`][`
10	`][][`
11	`[[]]`
12	`[-[-]-]---[-]`
13	`[][[[[[]]]Mu][]][][[K[De]G][vbh[]Y][SCqK[T]]kI[l][[P][][]]M]U[]CR[Q+].+r[]Dqi[]KQxEy[u[g>]][nPeb][][Rl[[zp][z[R[nBN[lKI.n[][E[[]][y]]gM.f-FZ[[[+[][UMnoM[.[OWHkxm]TO[z[>]]g]][eAy]J[FT][dG+O[][r[][[[UeSg[[[[Q[.]cn[]]]Y[[fS]x]zA]][CGB[X[X[ulv]]G[]Rz]Xoyf]wa[mc.[uJmX[[]pC]]]RR<[tMPK[]j]ET][w][-a[][Hpl[[]]]]]]]]]]]]]]]]`
14	`Rk[[]b.h][][+f][uk[F[]]]VO[][wamX[c[][D]KyHPkdQ[<Y[s[M]hr[]]]]]T[[J[r]S[[[[[[E]]]]][T[[[gN]D][[]]Dl]]n][vr][BO]eNl[][Zrs]PQf[[nn[]G]o]i[F>]eJMR[][]La[T][OwS][[].D[uJ][]][IC[-]][][[>+[[[[[]]]]][iw[]Jj]]]N<[]q[g[]g]x[TIup]n[Lv[[aE<ddT>WQjc[]p[]]ZZ<]RO][[vzv[R]wB.[[][fL][]i]v]]T[[r<Ih]vY[vw[]][En[]]o][[T]]`

15	[zf]oFIeB]fuJP-R+[uE[[]oj][]]][]o]ig>]][LhPjuM]]yA apcB.hP[e.o][]WOLub]]g][OjF]]R[[h[]]YKkxUV[mSZR[[[I]]z]Z[[]][[+][[[jLv]]][]][[R][[I[][[t]Hx]wY]r]]z[[[]]n[e+]g+vkh[]B[]h][YH[[e-]]W]r][P]][]]E][n[YY]O] pXh[]]<K[W[KZ[J][bKI[]]]v]SpMbSvUY[AF]]u]heC]Y][[q]]Mnas]]]E[[[Bj]qFrN]<]X]y[]][xj[[]]I[KQEap[TS][]N
16	DD[b]Y[[p.][Tv]]yv]]][am]Y][[dH>[][CV[TGx]Dkwi][]] R[LqL[][]]u]W[][]][V]M[]d][>]j]A]T]dgjY[u]]t]]u[]E N[[x[]]y[]Be[QUJcem][[-gfp]+[i]rG[>]N[[]]g]]b<[[>[K]]O[]]]YshKEG]]]]Sqt[]][pvpR]APV]P]s]]q]s]W[[OO+]]P][]]Eh[e[]][[]nI][h]qK[p][QrsXdP]vjl][t[JQ[ruSba k>[G[]U]]]t[-rp]]]r][]x[][JOAZ[<X[[O]]IW-H]]Z]N[][
17	]]K[][][[]AK]>]n[IP]<-VFG[Mh][Yofgy[jJip]PcQ[[<[dM]L][MU[[UK]]][.gY]M[D][tv][K]][eGIuG[]]vI]E[[]w[[Y [J[D>[-rn]E[[w]OX]v][Wp]P]]T]r[[]iX.qh>]pw][V[]][C []K[.[[[qz+OSV]yc[[[P[G[[[Phz][[E]]WI[][]X[]R]c][[Ec[[Gp[]ki]]v-t]]h]R[][qGbk+[][[[]v[r]]iN][j][]]][ ]O.[[Cx]][[PsGY][ez]a][][T-[]Ss[]E[[[[]Xc[i]R.k]L[ ]]]]]]]
18	[][[[[[]]]u][]][][[K[e]G][vh[]Y][SCqK[T]]kI[l][[P] [][]]M]U[]CR[Q+].+r[]Dqi[]KQxEy[u[g>]][nPeb][][Rl[[zp][z[R[nBN[lKI.n[][E[]][y]]gM.f-FZ[[[+[][UMnoM[.[OWHkxm]TO[z[>]]g]][eAy]J[FT][dG+O[][r[][[[UeSg[[[[Q[.]cn[]]]Y[[fS]x]zA]][CGB[X[X[ulv]]G[]Rz]Xoyf]w a[mc.[uJmX[[]pC]]]RR<[tMPK[]j]ET][w][-a[][[[]]]]]]]]bb]]]]aa]]]]

Problem 4 – Zombie Hordes

It's the zombie apocalypse and you are trying to seal yourself in a warehouse with several entrances. You've found that near each entry in the warehouse are piles of boxes of different sizes that you can use to seal the entry. Each of the boxes has the same height, which is exactly the height of the entry, but they have different widths. Through trial and deadly, horrible error, you've found that the best configuration is to put exactly three boxes side by side in an entry so that their combined width is exactly the width of the entry. When using the boxes, you can only use each box once.

For instance, near an entry of width 100 you have boxes of width 15, 20, 41, 44, 50, 60, 90, 100. In this situation, you can push the boxes of width 15, 41 and 44 into the entry to exactly fit the width.

Write a program that determines if there is a way to seal the entry or not, given its width and the width of the boxes near the entry.

Input

Each input file contains two lines: the first line contains one integer, giving the width of the entry. The second line has a list of integers, all less than or equal to the width of the entry, giving the widths of boxes that are available near the entry.

Output

Output `safe` if there is a combination of three boxes that fits in exactly in the entryway. If there is no combination that fits exactly, write nooo with three o's.

Sample Input 1	Sample Output 1
100 15 20 41 44 50 60 90 100	safe

Sample Input 2	Sample Output 2
31 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30	nooo

Judging Data for Problem 4

Case	Input
1	100 15 20 41 44 50 60 90 100
2	100 1 2 3 4
3	31 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
4	100 1 2 3 4 5 6 7 8 9 10
5	100 35 36 37 38 39 40 41 42 43 44 45 46
6	10 2 3 4
7	120 92 96 88 82 62 86 41 110 77 50 31 33 19 50 27 90 37 53 16 60 117
8	247 157 23 26 95 215 215 229 183 14 11
9	207 79 79 57 205 179 29 53 175 71 170 100 161 119 116 110 27 53 194 198 142 143 169 115 156 190 100 186 118 78 118 22 187 199 41 56 29 75 80 113 36 171 56 148 110 172 161
10	165 4 73 4 133 5 12 162 145 112 12 151 21 23 39 125 69 156 66 44 122 130 117 70 104 67 50 127 85 67 121 37 127 109 150 118 1 144 70 116 87 93 75 126 66 93 23 37 53
11	31 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
12	5 3 2

Problem 5 – Sudoku

Sudoku is a puzzle where numbers are placed in a grid. In this version, we use the numbers 1-4 and the grid has size 4x4. When completed successfully, a Sudoku puzzle has the numbers 1-4 appear one time in each row, one time in each column and one time in each of the four 2x2 squares in the corners of the grid. For instance, the following shows a successful and unsuccessful Sudoku puzzle:

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	3

1	2	3	4
3	4	1	2
2	3	4	1
4	1	2	4

As you can see, the puzzle on the left is completed successfully: each row, column and square (defined by the bold lines) has the numbers 1-4 exactly once. However, the puzzle on the right is not successful: it contains the number 4 twice in (for example) the bottom row of the puzzle. The number four is an error.

Write a program that reads in sixteen numbers from 1-4 and determines if the puzzle was successfully completed or not. If the Sudoku puzzle was successful, write “good”. But if the puzzle is not successful, report the first duplicate value in the puzzle. A duplicate value is any entry that has the same value above it, to the left of it, or in the same square with it (when read left-to-right, then top-to-bottom, i.e., in English reading order). If there are several duplicates, report the first one in left-to-right, top-to-bottom order.

1	2	3	4
3	**3**	1	2
1	2	4	3
4	3	2	1

For instance, in the incorrect puzzle on the right above, the first duplicate is the 4 in the bottom-left hand corner: it is a duplicate in its column, its row, as well as in the bottom-right 4x4 square. In the puzzle on the right, the first duplicate is the 3 in the second row and second column (shown in bold). This is a duplicate of 3 in the second row, as well as in the upper-left hand square of the puzzle. Note that the 3 to the left is not a duplicate in the left-to-right top-to-bottom order, and that there are other duplicates in this puzzle.

Input

The input consists of four lines of integers, each with four integers on each line. The integers will be values from 1 to 4.

Output

The output for the program should either be the word “good” (without quotation marks) or two integers, giving the row and column of the first duplicate value. The

row and column numbers should be between 0 and 3 (inclusive): the top row is row number 0 and the left-most column is column number 0.

Sample Input 1	Sample Output 1
1 2 3 4 3 4 1 2 2 3 4 1 4 1 2 3	good

Sample Input 2	Sample Output 2
1 2 3 4 4 1 2 3 3 4 1 2 2 3 4 1	1 1

Judging Data for Problem 5

Case	Input
1	1 2 3 4 3 4 1 2 2 3 4 1 4 1 2 3
2	1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
3	1 2 3 4 2 3 4 1 3 4 1 2 4 1 2 3
4	1 2 3 4 4 1 2 3 3 4 1 2 2 3 4 1
5	1 2 3 1 1 2 3 1 1 2 3 1 1 2 3 1
6	1 1 3 4 2 3 4 1 3 4 1 2 2 1 4 3
7	4 1 2 3 2 3 1 4 3 2 4 1 1 4 3 4
8	3 1 4 2 2 4 3 1 1 4 2 4 4 2 1 3
9	3 4 1 2 1 2 3 4 2 3 4 1 4 1 2 3

10	3 4 1 2 1 2 3 4 4 3 2 1 2 1 4 3
11	4 3 2 1 2 1 3 4 1 2 4 2 3 4 1 2
12	2 1 4 3 4 3 1 2 3 4 2 1 4 2 3 4

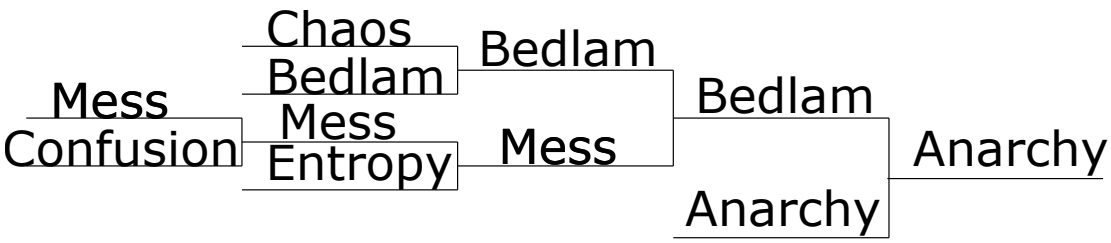
Problem 6 – DHL Playoffs

You are the organizer of a local hockey league called the DHL - the Disorganized Hockey League. The league is a place for people that are really, really disorganized people to be involved in a league sport. And after a painfully long season, it's time for playoffs! Unfortunately, the playoffs have been especially difficult for you to coordinate, as the playoffs are during your holidays and the games have been scheduled by the teams.

The playoffs have been scheduled as follows: teams randomly show up to the rink to play games. If two teams are there at the same time, they play a game. When two teams play, they record the result on a white board and the losing team goes home: they are out of the playoffs. The winning team comes back on another day to play another game with another team that is still in the playoffs. This continued until one team was left, who is the winner of the league.

You have reconstructed the playoff brackets, that is, the list of who played who to give the champions, but because of the organization of the playoffs, some teams played fewer games than others in the playoffs. You would like to calculate the hardest path in the playoffs: given how the playoffs happened, what is the maximum number of games a team would have had to win to be champions. Be careful: we're not asking what the maximum number of games any team won in the playoff brackets. We're asking: "if we think of how many games each team would have had to win to be champions in the bracket, what is the maximum number for all the teams?"

For instance, consider the following playoff bracket:



In this bracket, the Anarchy are the champs, but both the Mess and the Confusion had the hardest path: they would have had to win four games to win the DHL championship.

Input

The input for the problem is given as a list of games. Each part of the bracket in the playoff has been numbered and each line of input describes a game. The lines can have one of two forms:

1. "a X b c" where a,b and c are integers and X is a team name. This says that "in game a, the winners of games number b and c played and the

winner was team X". The team name is a string (no spaces) of at most 16 characters (letters and numbers). The bracket numbers b and c are guaranteed to appear *before* this line in the input.

2. "a X" where a is an integer and X is a team name. This line says "in position a is team X". The team names are strings of characters (no spaces) of length at most 16. This represents a team that is playing in a "first round" playoff game (i.e., this represents a team that is about to play its first playoff game).

The ***last*** line of the input gives the result of the championship game (i.e., the team listed in the last line of the input is the champion).

Output

Output a single integer z representing the worst case number of wins any team would have to have in the playoff structure to be champions.

Sample Input 1	Sample Output 1
11 Confusion 10 Mess 9 Bedlam 8 Chaos 7 Entropy 6 Mess 10 11 5 Mess 6 7 4 Bedlam 8 9 3 Bedlam 4 5 2 Anarchy 1 Anarchy 2 3	4

Sample Input 2	Sample Output 2
1 Giraffes 2 Dogs 3 Iguanas 4 Jellyfish 5 Dogs 1 2 6 Iguanas 3 4 7 Dogs 5 6 8 Eels 9 Dogs 7 8 10 Alligators 11 Alligators 9 10 12 Cats 13 Alligators 11 12 14 Bears 15 Alligators 13 14	6

Judging Data for Problem 6

Case	Input
1	11 Confusion 10 Mess 9 Bedlam 8 Chaos 7 Entropy 6 Mess 10 11 5 Mess 6 7 4 Bedlam 8 9 3 Bedlam 4 5 2 Anarchy 1 Anarchy 2 3
2	1 Giraffes 2 Dogs 3 Iguanas 4 Jellyfish 5 Dogs 1 2 6 Iguanas 3 4 7 Dogs 5 6 8 Eels 9 Dogs 7 8 10 Alligators 11 Alligators 9 10 12 Cats 13 Alligators 11 12 14 Bears 15 Alligators 13 14
3	10 name10 6 name6 34 name10 10 6 23 name23 38 name10 34 23 13 name13 14 name14 33 name13 13 14 39 name10 38 33 9 name9 45 name10 39 9 21 name21 17 name17 2 name2 24 name17 17 2 18 name18 26 name17 24 18 3 name3 19 name19 27 name3 3 19 30 name17 26 27 37 name21 21 30 20 name20 0 name0 35 name20 20 0 *(case continues on next page)*

	43 name21 37 35 8 name8 4 name4 22 name22 25 name4 4 22 15 name15 29 name4 25 15 31 name8 8 29 16 name16 5 name5 32 name16 16 5 1 name1 11 name11 28 name1 1 11 40 name16 32 28 41 name8 31 40 12 name12 7 name7 36 name12 12 7 42 name8 41 36 44 name21 43 42 46 name10 45 44
4	13 Team13 33 Team33 37 Team37 41 Team41 18 Team18 67 Team41 41 18 27 Team27 72 Team41 67 27 73 Team37 37 72 77 Team33 33 73 81 Team13 13 77 16 Team16 17 Team17 50 Team16 16 17 26 Team26 57 Team16 50 26 28 Team28 43 Team43 9 Team9 87 Team43 43 9 22 Team22 4 Team4 20 Team20 51 Team4 4 20 54 Team22 22 51 88 Team43 87 54 89 Team28 28 88 90 Team16 57 89 19 Team19 25 Team25 58 Team19 19 25 10 Team10 59 Team19 58 10 45 Team45 74 Team19 59 45 1 Team1 75 Team19 74 1 *(case continues on next page)*

	48 Team48 79 Team19 75 48 92 Team16 90 79 30 Team30 38 Team38 65 Team30 30 38 8 Team8 82 Team30 65 8 46 Team46 83 Team30 82 46 35 Team35 32 Team32 40 Team40 61 Team32 32 40 76 Team35 35 61 24 Team24 6 Team6 69 Team24 24 6 80 Team35 76 69 49 Team49 85 Team35 80 49 86 Team30 83 85 0 Team0 91 Team30 86 0 94 Team16 92 91 7 Team7 21 Team21 68 Team7 7 21 34 Team34 44 Team44 5 Team5 55 Team44 44 5 12 Team12 47 Team47 62 Team12 12 47 70 Team44 55 62 11 Team11 78 Team44 70 11 84 Team34 34 78 93 Team7 68 84 96 Team16 94 93 29 Team29 42 Team42 31 Team31 53 Team42 42 31 15 Team15 60 Team42 53 15 63 Team29 29 60 36 Team36 64 Team29 63 36 14 Team14 23 Team23 52 Team14 14 23 66 Team29 64 52 2 Team2 71 Team29 66 2 3 Team3 39 Team39 56 Team3 3 39 *(case continues on next page)*

	95 Team29 71 56 97 Team16 96 95 98 Team13 81 97
5	6 Team6 28 Team28 14 Team14 49 Team28 28 14 51 Team6 6 49 33 Team33 12 Team12 61 Team33 33 12 30 Team30 35 Team35 56 Team30 30 35 19 Team19 13 Team13 52 Team19 19 13 59 Team30 56 52 27 Team27 60 Team30 59 27 63 Team33 61 60 64 Team6 51 63 4 Team4 68 Team6 64 4 1 Team1 31 Team31 47 Team1 1 31 26 Team26 3 Team3 42 Team26 26 3 54 Team1 47 42 9 Team9 69 Team1 54 9 2 Team2 0 Team0 25 Team25 10 Team10 38 Team25 25 10 41 Team0 0 38 44 Team2 2 41 29 Team29 58 Team2 44 29 16 Team16 24 Team24 48 Team16 16 24 18 Team18 20 Team20 32 Team32 39 Team20 20 32 46 Team18 18 39 62 Team16 48 46 7 Team7 5 Team5 22 Team22 43 Team5 5 22 50 Team7 7 43 15 Team15 65 Team7 50 15 66 Team16 62 65 *(case continues on next page)*

	67 Team2 58 66 11 Team11 36 Team36 45 Team11 11 36 23 Team23 21 Team21 37 Team23 23 21 17 Team17 53 Team23 37 17 55 Team11 45 53 34 Team34 8 Team8 40 Team34 34 8 57 Team11 55 40 70 Team2 67 57 71 Team1 69 70 72 Team6 68 71
6	1 Team1 2 Team2 3 Team1 1 2
7	63 Team63 62 Team62 61 Team61 60 Team60 59 Team59 58 Team58 57 Team57 56 Team56 55 Team55 54 Team54 53 Team53 52 Team52 51 Team51 50 Team50 49 Team49 48 Team48 47 Team47 46 Team46 45 Team45 44 Team44 43 Team43 42 Team42 41 Team41 40 Team40 39 Team39 38 Team38 37 Team37 36 Team36 35 Team35 34 Team34 33 Team33 32 Team32 31 Team62 62 63 30 Team60 60 61 29 Team58 58 59 28 Team56 56 57 27 Team54 54 55 26 Team52 52 53 25 Team50 50 51 *(case continues on next page)*

	24 Team48 48 49 23 Team46 46 47 22 Team44 44 45 21 Team42 42 43 20 Team40 40 41 19 Team38 38 39 18 Team36 36 37 17 Team34 34 35 16 Team32 32 33 15 Team60 30 31 14 Team56 28 29 13 Team52 26 27 12 Team48 24 25 11 Team44 22 23 10 Team40 20 21 9 Team36 18 19 8 Team32 16 17 7 Team56 14 15 6 Team48 12 13 5 Team40 10 11 4 Team32 8 9 3 Team48 6 7 2 Team32 4 5 1 Team32 2 3
8	61 Jets 60 Ducks 59 Jets 60 61 58 Coyotes 57 Jets 58 59 56 Bruins 55 Jets 56 57 54 Sabres 53 Jets 54 55 52 Flames 51 Jets 52 53 50 Hurricanes 49 Jets 50 51 48 Blackhawks 47 Jets 48 49 46 Avalanche 45 Jets 46 47 44 Jackets 43 Jets 44 45 42 Stars 41 Jets 42 43 40 Wings 39 Jets 40 41 38 Oilers 37 Jets 38 39 36 Panthers 35 Jets 36 37 34 Kings 33 Jets 34 35 32 Wild 31 Jets 32 33 30 Canadiens 29 Jets 30 31 28 Predators 27 Jets 28 29 *(case continues on next page)*

	26 Devils 25 Jets 26 27 24 Islanders 23 Jets 24 25 22 Rangers 21 Jets 22 23 20 Senators 19 Jets 20 21 18 Flyers 17 Jets 18 19 16 Penguins 15 Jets 16 17 14 Sharks 13 Jets 14 15 12 Blues 11 Jets 12 13 10 Lightning 9 Jets 10 11 8 Leafs 7 Jets 8 9 6 Canucks 5 Jets 6 7 4 Knights 3 Jets 4 5 2 Capitals 1 Jets 2 3