Course: CpSc 212: Algorithms and Data Structures Spring 2014
Lab 03: Integer Set Data Structure Using Sorted Linked Lists

1 The Integer Set Data Structure

In this lab, you will be building an integer set data structure using sorted linked lists. Remember that a *set* is a collection of distinct elements. We will be using a sorted linked list to represent a set of integers.

1.1 Downloading and Extraction

Please first download the *lab03.tar.gz* file from the following webpage:

```
http://people.cs.clemson.edu/~rmohan/course/cpsc2120-sp14/
```

Next, please extract the files using the following command.

```
tar -zxvf lab03.tar.gz
```

This command will extract all the files in the directory named *lab03*. Please change your directory using the command:

```
cd lab03
```

1.2 IntSet specifications

Please open *intset.h* to see the specifications of the class. The class *Node* represents a node in the linked list. It contains the integer data *key* and a pointer to the next node in the linked list. Note the public interface to the *IntSet* class. The pointer *head* points to the first element in the linked list. The linked list should be sorted at all times. Open intset.cpp and implement the following functions.

1. ***find(key):*** This function returns *true* if the key is present in the set. Otherwise returns *false*.
2. ***insert(key):*** This function inserts an integer key into the set. It is an error to call this function with a value of a key that is already present in the set.
3. ***remove(key):*** This function removes a key from the set. It is an error to call this function with a key not present in the set.
4. ***print():*** This prints the contents of the set in increasing order.

The constructors and the destructors for the *IntSet* class are already provided for you. We currently use a "dummy" node to represent the begining of the list. Please feel free to change the constructors and destructor to suit your implementation of a linked list. Please note the *assert* statement at the begining of funcitons *insert* and *remove*. The *assert* statement allows execution of the next line of code only if its corresponding condition is *true*.

1.3 Testing your IntSet class

You can use *testIntSet.cpp* or create your own test program to test your IntSet class. The file *testIntSetInput* contains a list of integers that can be used to insert into the set. Please compile your test program as:

g++ testIntSet.cpp -o testIntSet intset.cpp

You can then execute using the command:

./testIntSet < testIntSetInput

Also, uncomment certain lines to check if your integer set operates correctly on inserting duplicate elements and removing absent elements. Make sure to recompile your program every time you make changes to *testIntSet.cpp*.

2 Jolly Sequence

A sequence of $n > 0$ integers is called jolly if the absolute values of the difference between successive elements take on all the values 1 through $n - 1$. For example,

3 11 15 6 7 5 −2 3 −3 −6

is a jolly sequence. The absolute values of the difference between successive elements are 8, 4, 9, 1, 2, 7, 5, 6 and 3 respectively, which make numbers 1 through 9. The definition implies that any sequence of a single integer is jolly. Please write a program to determine whether or not a sequence is jolly.

2.1 Input

The input contains the total number of integers n, $1 \leq n \leq 1000$ in the first line followed by the n integers separated by a space from the second line.

2.2 Output

Print "Jolly" if the corresponding sequence is a jolly sequence. Otherwise print "Not Jolly".

Sample I/O

Test case#	Input	Output
1.	10 3 11 15 6 7 5 -2 3 -3 -6	Jolly
2.	10 3 11 15 6 7 5 -2 3 -3 -5	Not Jolly

2.3 Compiling and Testing

You are to write a stand-alone program that determines whether a sequence is jolly or not. Please name your program as *jolly.cpp*. This program **should** make use of the integer set data structure you implemented in the previous section. Please compile your program using the following command:

g++ jolly.cpp -o jolly intset.cpp

You can put your input in a file named *in* and then execute your jolly program using the command:

./jolly < in

Please thoroughly test your program and make sure it works for all cases.

2.4 Additional Reporting

The files *input1* and *input2* contain a sequence of 1000 integers. Please execute your jolly program by taking input from these files. You can redirect the input and execute your program using the command:

./jolly < input1

Please report your outputs in the file *output*.

3 Submission

Please submit your solutions through handin.cs.clemson.edu. The due date for this assignment is Friday February 7th at 11:59pm.

4 Grading

Each section in this assignment is graded out of ten points. No points will be awarded to a program that does not compile. You will recieve 1 point for successful compilation and 1 point for successful execution (no seg faults and infinite loops). 3 points is awarded for correctness and efficiency each and 2 points awarded for good code: readability, comments, simplicity and elegance, etc.