README: Write a program that implements the work life at the North Pole.

Santa1
Initial instructions
Part A. Santa 1 Keep Santa Alive, let him rest

a) When Santa checks the workList and there is no work, he waits (blocks) until some work is created and added to the workList. To wake Santa up, the UI thread needs to do a notify each time work gets added to the workList. Remove the lines where Santa sleeps. We are in effect, replacing sleep with notify and wait.

b) The data (ArrayList) is not thread safe. Make it thread safe.

c) When Santa dies from loss of power, the UI thread is still alive and the program keeps running.

Modify the program so that when Santa dies of power loss, the UI thread is terminated.

d) When you terminate the UI thread by typing “quit”, Santa is still alive and the program keeps running. Modify the program so that when the UI thread dies, the Santa thread is terminated (i.e. you kill Santa / )

e) To set up the next part of this questions, modify the UI code so that you can enter multiple tasks separated by commas. Thus, if the user enters: build sleigh, fix toy, make game there should be three tasks added to Santa’s work list.

Santa2
Initial instructions
Part B. Santa needs Three Jobs to Wake up.

In order not to run Santa ragged, waking up every time an elf needs help, modify your code so that we ONLY wake Santa up when there are three or more tasks waiting in the queue.

• When Santa awakes, he does each task (using up 1 power for each) and then goes back to sleep. No tasks should be added to the work queue until Santa has finished all the jobs in the queue.

• Create a new version of your program, call it Santa2.java. For ease of testing and deployment, include additional classes in one file. When you do this, you must only have ONE public class. For the rest, leave the word public off the class definition. These now become default package classes.

Santa_cpp
Implementation of the Santa2 program in C/C++ using the pthreads library