Group 4: Isaac Supeene and Braeden Soetaert

What is your definition of an object?
Any old Ruby object. All the objects should be comparable with each other.

What strategies should be deployed in terms of accepting input (i.e. the large number of objects.)?
Unless ‘large’ means ‘over 500 million’, an array will be fine.

Is your sort generic? What sorting criterion does your system use? Is this criterion flexible; i.e. changeable by the user at the point of execution? What limitations have you placed upon legal sorting criteria? To make it reusable to other people, how should we include it into the Ruby Class hierarchy?
By default, we will use the ‘<’ operator to compare elements. We will allow the user to pass in their own comparison function, and specify whether the sort should be ascending or descending.

In reality we have a uniprocessor system, describe “equationally” what is happening to your solution as you increase the concurrency (e.g. produce a regression model (remember regression from Statistics) of processing time against the number of threads. Your solution can be modeled as a program which: (1) has a component which produces threads; and (2) a set of threads which undertake the same (small) task. This is in essence the basis of stress testing (discussed in ECE 322)
The distribution that best represents this problem would be a bathtub curve or a parabola. This is because as we increase the number of threads, initially it will lower our processing time because we can execute the small tasks in parallel. At some point though the increased number of threads will cause processing time to start increasing due to the number of context switches and time to create threads. Additionally we are running on a uniprocessor, so we can only execute so many threads in parallel at once due to hardware constraints so additional threads end up increasing cpu time.
Y = processing time
X = number of threads. Including main thread so min $X = 1$
P = number of threads that can run concurrently on our uniprocessor system. Should be a constant as we don’t have threads that block waiting for IO.
β = overhead of creating thread as well as context switches
TT = non-concurrent task time
Regression Model:
$Y = [TT/\min(P,X)] + \beta*(X-1)$

or Y decreases greatly as X increases up to P and then Y increases slowly as X grows larger than P. X-1 to account for only the main thread (X = 1).

Concurrent systems tend to crash frequently – what approach to exception-handling have you devised? Consider the content of the library at: http://c2.com/cgi/wiki?ExceptionPatterns; which are applicable to this problem?
Is Module Errno useful in this problem? What components of the Ruby exception hierarchy are applicable to this problem? Discuss in detail your strategy for exception-handling.
We believe many of the exception patterns listed on that site are useful to most problems but the most useful ones for this exercise are the following: Tidy Up Before Throwing, Exceptions Tidy Threads, and Cancel threads with an exception. These ones are the most relevant because we will be creating threads and our exceptions are most likely to happen either in the creation of these threads or inside the threads themselves. No, Errno is not useful because we will not be making system calls. One important part of the ruby exception hierarchy here is the ThreadError due to us working with threads and if something goes wrong with them. Another is the NoMemoryError as we will be spawning many threads and although threads are not as expensive as forking, we may still run into memory issues if the number of objects is big enough or the system hardware isn't good enough. Our main strategy for exception handling will be to notice exceptions that are raised in our threads and to retry the work that thread was to do if the thread generated an exception. If the thread generates an exception again then our program will return an error and not sort the array of objects.

What differences exist between thread-based and process-based solutions? How has this impacted the design of your solution?
A process-based solution would require inter-process communication, meaning that it's not practical to hold the state in memory. Since we plan to take input in an array, obviously a single process is the way to go in this case.

Do you have any race-condition or task synchronization concerns about your solution? How do we tidy-up a multi-threaded program, if stopped mid-execution?
Yes! We need all the threads to terminate after the specified duration, and return control to the sender. That means we can't guarantee that the list is completely sorted when we return, but we should at least guarantee that the list contains the same elements as before! This means that swapping two elements is a critical section. If a thread is in the middle of swapping two elements, it needs to finish before it shuts down.

To allow each thread to finish what it needs to do, we should employ a java-style cooperative threading model, where each thread has an opportunity to finish up what it needs to do before stopping.

As discussed in ECE 320: What is configuration management? What is version control? Are you using either concept? If "yes", describe your process and any tool support what you utilize – illustrate your process with regard to Assignments 1 and 2; if "no", justify your decision.
With regards to software, configuration management is the tracking and controlling of changes in the software. Configuration management can involve version control but also relates to overall management such as teamwork, defect tracking, and making sure every important component is present. Version control is tracking changes to the source code so they can be easily reviewed, merged and if necessary, reverted. We are using these concepts by utilizing Github for our repository needs. Through Github we are creating separate branches for each change and then having the other partner review the code before we pull those changes into our master branch. This process allows us to try and minimize defects because two sets of eyes have looked over each contribution. Also this process allows us to coordinate well and stay up to date on what the other person is doing.

Briefly Explain:
a. What is refactoring (as discussed in ECE 320)?
b. Are you using refactoring in your development process? Justify your answer?
c. If "yes", give examples, minimum of 2, of the refactoring "patterns" that you used in Assignment 1
d. If "no", give examples of where your solution to Assignment 1 would be improved by applying refactoring patterns. Supply a minimum of two different (i.e. different refactoring patterns) as examples.
Refactoring is changes to the code which do not affect its functionality, but improve maintainability attributes such as readability, testability and extensibility.
Examples of refactoring in assignment 1:

1. Factoring out common matrix functions into a module. Several of the matrix functions are common to sparse matrices and tridiagonal matrices, so we factored them out into a separate module which was included in both classes.
2. Factored out symbol related code from our contracts module. Our contracts module dynamically added the preconditions and postconditions and invariants to all the functions, and some of this involved converting between contract method names, and regular method names. The symbol manipulation was separable from the main

contract code, so we split it out into a ContractSymbols module. This turned out to be a great decision, since when we completely altered the method of including the contracts for performance reasons, we were able to easily reuse the ContractSymbols module.