1. Identify the problem.

The assignment given is an archiving challenge. We are to construct a program that takes a specified directory and archives it to a file of a user-specified name. We then construct a program that will unarchive the file and restore the directory, potentially in a different location. The general problems for the archive portion are to figure out how to get the necessary information for file and directory reconstruction, such as file names, file sizes, number of files, and the name of the directory, as well as ensuring every file is being written to the archive file in its entirety. For the unarchive portion, the challenge is going to involve ensuring that the new directory is created, the correct number of files are written, the files are the correct size, and the files contain their original content. General things to consider when writing the archive code are to ensure that the directory exists and handle the situation if it doesn’t, handling an empty directory, handling empty files or files that have white space in them, the file extension, if the program is being run with too few or too many arguments, and potentially hidden files. For the unarchive program, general consideration are making sure the archive file exists, checking to see if the directory in the file already exists and how to handle it, and ensuring that files are being created and written to the correct file name, format, and content.

1. Identify solutions.

All of the solutions for this assignment/problem are related to bookkeeping. Keeping track of the directory name, the file names, file sizes, and number of files is less important for archiving but incredibly important for unarchiving. The general steps to follow are to record the directory name, the number of files, the file sizes, and file names. After recording the metadata create a new file which will hold all the archive data, then read and write the files to the archive file. A very quick solution to the problem would be to walk through the target directory and write metadata to the archive file followed by the file contents. For this to work though, there would have to be some sort of distinguishing character or string that differentiates each file. For the unarchiving portion, files would have to be created with the data stored before the content, then the content written itself. Another solution would be to make multiple structures for each file’s metadata, then write the contents of the file below it. The unarchiving portion for this solution would rely on the data in the structure before the content of the file. A third solution would involve making one structure for the metadata and using that data to reconstruct the files after unarchive is invoked. This would be a little more complicated than the previous solution in that the structure would have to be accessed for each new file to be created.

1. Evaluate solutions.

Out of the three solutions, I think that both solutions involving the structures are ideal. The first solution involves writing everything to the archive file without much thought into how things are organized. When unarchiving after this solution, there’s going to be a lot of things to keep track of to ensure files are created properly. For the two structure-based solutions, keeping track of data is going to be much easier. However, the big difference in the two is the first solution is more dynamic than the second. However, the second solution would take up less space than the first. For the purpose of this assignment, I will be attempting the second structure solution; one structure at the start of the archive file containing all the metadata for the archive file. I’ll keep it simple and prepare it for a good number of files. If I find I’m having issues, I can easily switch to the solution of placing a structure before a files contents.