TCES 202 Advanced Programming
Statistical Package Project
Phase – I

Background

This project will give you a first glimpse of building a program with many parts that must be integrated into whole package. The project will be conducted in three stages. This will give you a sense of how real live projects are conducted. You will be working in pairs (called pair programming), which has been found to maximize student learning.

The project will be to build an application that computes (simple) statistics for a sample of data. In this case the package supports floating point data (actually stored as doubles to minimize round-off error). The data will be entered into a data file and the user can direct the application to manipulate the data and compute the statistics. For this version we will stick with very simple statistics of the data set, the mean, median, variance, and standard deviation. For those who are interested in extending this project it is also possible to learn how to plot distributions and incorporate several statistical tests, but that is not part of the requirements. Useful resource at http://www.chem.utoronto.ca/coursenotes/analsci/StatsTutorial/MeasMeanVar.html

Product Description

StatPac is a simple package that allows a user to capture a set of data, as real numbers in floating point format, manage the data set (e.g. adding more data, editing data, etc.) in files, and then having the program compute and update the statistics for that data set. The program can display the data on the screen or save to a formatted file for printing. It will also provide reports on the statistics computed.

The program is what we call menu-driven. In other words, the user is presented with a menu of choices (see below) for actions to be taken. This is exactly what most graphical user interface programs, like Windows applications such as Notepad do. However, this program will be a non-graphical (that is text-based) version similar to what was done before the advent of modern graphics. As a matter of fact, it would be possible to port the code you will write to a C++ program to add a graphic user interface (GUI) to replace the menu.

Learning Objectives

This project is intended to help you achieve the following learning objectives:

1. How to write modular programs (multiple .c and .h files that are linked together)
2. How to follow a design pattern in planning and implementing the code
3. How to do simple unit testing of individual modules and functions

4. How to incrementally develop and test and then integrate a system
5. What is involved in developing a full application
6. How to work in pairs to learn and produce working applications (later in the series you will work in teams)
7. (C objectives) Working with pointers, heap memory dynamic allocation, standard library functions

These are extremely important objectives and what you learn here will be used in many of your future courses.

Product Specification

I am acting as the chief product designer (in future courses you will learn more about designing complex systems). I have developed a functional specification for this product that you will use as the basis for developing the system. This is a fairly simple and straightforward specification but you will find the implementation and testing will be a challenge for your current level of skill at programming.

The product to be produced shall conform to the following.

Overview

The product will have the following general functionality.

```
StatPac - Statistical Package
Routines for capturing and storing data, computing statistics, and generating
reports

Program

Main Menu       Sub-menus
        File
                Create new data file
                Open an existing file
                Rename file
                Copy file to...
                Print file contents
                Delete file
        Edit
                Add data
                Edit item
                Delete item
        Search
                Search for item
                Search and edit
                Sort data
        Compute
                Update all
                Compute mean
                Compute median
                Compute variance
```

```
                Compute standard deviation
        Report
                Display statistics
                Print statistics
```

Files and Functions

Below is the list of .c files that will be created along with the major functions that will be built into those files. The first three items in the list have "man" (manual) pages that provide details as required on each object. Use these files as examples of how to produce external documents for the application. There will be a man page for each major file and function.

```
statpac.c
        main() prints menu, gets user selections, calls appropriate submenu
functions.
        main_menu()

file.c
        file_menu() prints submenu, gets user selections, calls appropriate
functions
        file_create() gets user input for new file name, checks to make sure
no file of that type already
           exists, if it doesn't exist it creates the file and initializes the
header lines (see below)
        file_open() opens an existing file if it can
        file_rename() allows the user to change the name of an existing file
to an alternate
        file_copy() makes a copy of an existing file. The new file must have a
different legal filename
        file_print() copies a data file to a formatted version suitable for
printing on a printer
        file_delete() allows the user to delete an existing file; gives the
user a second chance to abort

edit.c
        edit_menu() prints the submenu for the edit functions
        edit_add() allows a user to add addition data records to an existing
file
        edit_edit() allows a user to edit an existing data record
        edit_delete() allows a user to delete an existing data record

search.c
        search_menu() prints the submenu for searching a data file
        search_item() allows the user to find a specific data record if it
exists
        search_sort() allows users to sort the data file by data values, entry
or edit dates

compute.c
        compute_menu() prints a menu of computing options
        compute_update() allows the user to update the computations of all
statistics
        compute_mean() computes the mean value only - displays it and inserts
it into the header record
```

```
        compute_median() same for the median - if the file is not sorted, will
sort an internal array of data
        compute_variance() same for variance if mean is updated
        compute_stdev() same for std. dev if mean and variance are updated.

report.c
        report_menu() prints report submenu
        report_display() shows statistics report on the screen
        report_print() prints the report to a file for printing
```

Data File Format

The data file has a particular structure that contains meta-data (data about data) to support file and data management. The file consists of a "header record", the first line of a text file, and some number of data records. The records are stored on disk in what is known as comma separated format (CSF). The two kinds of records are shown here with respect to the semantics of their contents and their definitions in the main header file,

```
statpac.h.

Header Record
        Filename
        Date of creation
        Date of last modification
        Owner name
        Number of records
        State of order (new, entry, sorted)
        Mean
        Median
        Variance
        Std. Deviation


Data Record
        Date of entry
        Date of modification
        Data

/* global.h - partial view */

typedef enum{new, entry, sorted} State ; // defines a State type for use
below

typedef struct file_header {
        char * filename;
        char creation_date[25];
        char modification_date[25];
        char * owner;
        long num_records;
        State state;
        double mean;
        double median;
        double variance;
        double std_dev;
} FileHeader;
```

```
typedef FileHeader *FileHeaderPtr;    // defines a pointer type of dynamic
allocation

typedef struct data_record {
        char entry_date[25];
        char mod_date[25];
        double data;
} DataRecord;

typedef DataRecord *DataRecordPtr;    // defines a pointer type of dynamic
allocation
```

File example: dataFile.dat

```
dataFile.dat,Sat Jan 10 11:15:06 2015,Sun Jan 11 11:15:06
2015,mmuppa,75,entry,0.0,0.0,0.0,0.0
Sun Jan 11 11:15:06 2015,Sun Jan 11 11:15:06 2015,345.67
Sun Jan 11 11:15:06 2015,Sun Jan 11 11:15:06 2015,222.22
  \
  /
  \
  /
Sun Jan 11 11:15:06 2015,Sun Jan 11 11:15:06 2015,541.01
```

The top line (wrapped) is the header record containing all of the meta-data and last computed statistics. The next 75 lines contain the actual data along with the entry and edit dates. All of the elements (called fields) are separated by commas. When reading the data from a file you need to parse the line, extracting each field into its own data variable in one of the above structures. The last line is actually not a printed line, it is the end-of-file marker, appended by the operating system when the file is closed after writing.

Project Organization and Requirements

All students will be paired (one team of two) for this project. The project will be done in three stages. The first stage will involve developing the main() function and the main_menu() function.

Phase I

Set up the files (.h and .c) indicated above. Develop the main() function in statpac.c and the main_menu() function. In each of the menu-related files, start with a stub function for that submenu. E.g. in file.c build a stub called file_menu() that simply returns.

The objective of phase 1 is to complete the implementation of all of the file management functions.

```
statpac.c
        main() prints menu, gets user selections, calls appropriate submenu
functions.
        main_menu()

file.c
        file_menu() prints submenu, gets user selections, calls appropriate
functions
        file_create() gets user input for new file name, checks to make sure
no file of that type already
           exists, if it doesn't exist it creates the file and initializes the
header lines (see below)
        file_open() opens an existing file if it can
        file_rename() allows the user to change the name of an existing file
to an alternate
        file_copy() makes a copy of an existing file. The new file must have a
different legal filename
        file_print() copies a data file to a formatted version suitable for
printing on a printer
        file_delete() allows the user to delete an existing file; gives the
user a second chance to abort
```

Submit your Phase I deliverables as a zip file on Canvas. Each file must have your names and the appropriate documentation in the header and for each function. Use the starter code provided for the header files. You must comply with the naming conventions provided.