1_1a

Command to Run

```
@make
```

Make File Break Down

This line creates the output generated by the preprocessor which has a .i extension. This file has all headers included, comments removed and macros replaced

```
@gcc -E 1a.c -o 1a.i
```

This line generates the output assembly file. Note this is pure assembly which is denoted by the lower case s in the extension '.s'. A capital '.S' indicates an assembly code which still needs to go through a preprocessor.

```
@gcc -S 1a.i -o 1a.s
```

The subsequent 3 lines generate and link the object file (.o) which then produces the executable which is run

```
@gcc -c 1a.s -o 1a.o
@gcc 1a.o -o 1a
@./1a
```

General Code Breakdown

- The countrows() function firstly counts the rows using open and read syscalls.
- The row count is then used to make the code workable for arbitary number of rows provided they follow the same format
- The struct marks_structure is used to store the assignment scores.
- The fork sys call then creates two processes. The child process has id = 0 which is used for section A otherwise the if else statement proceeds to compute for section B.
- In the if block -
  - The processData function is called which populates the 2D Marks Array and 1D section array with Data from the CSV.
  - Then the section variable is set to "A" indicating the function will be called for section A as this variable is used inside the printforgivensection function. The function computes the scores, stores and prints it.

  - For printing the stringPrinter function is used which takes the given format and prints it accordingly
  - After completion and exit(0) call is made to exit the child process.
- In the else block -
  - The processData function is called which populates the 2D Marks Array and 1D section array with Data from the CSV.
  - Then the section variable is set to "B" indicating the function will be called for section A as this variable is used inside the printforgivensection function. The function computes the scores, stores and prints it.
  - waitpid syscall is used to wait for a particular process. It's parameters are the process id to wait for, the status and an options parameter. By default, waitpid() waits only for terminated children. The stat stores the status incase of errors. If the stat variable equals EINTR it indicates -

    > WNOHANG was not set and an unblocked signal or a SIGCHLD was caught

  - In such a case a perror is raised and EX_SOFTWARE sysexit is used.
  - For printing the stringPrinter function is used.
- After this the program ends

Function Specific Breakdown (Including Error Handling) -

- countRows()

  - Open syscall is used to open the csv. The parameters are the file name and the oflag. From the documentation -

    > Values for oflag are constructed by a bitwise-inclusive OR of flags from the following list, defined in <fcntl.h> O_RDONLY specifies the file is Read Only since we do need to do any editing this is the perfect choice.

  - The return value is stored in a int variable file descriptor which is then used to check for errors. A -1 value stored indicates error while reading and hence an perror is raised.
  - Some variables are initialized for the process of reading and a read syscall is made with the arguments filedescriptor denoting the file which needs to be read, a buffer size which counts which specifies how many bytes from the file should be read and the count denotes the count parameter.
  - The errors are checked if the returned value of read is less than 0 a perror is raised.
  - Subsequently the program counts the lines using a while loop and returns the value while also storing it.

- processData(char* sectionTracker[rows-1], int marks[rows-1][6], int r)

  - Open syscall is used to open the csv. The parameters are the file name and the oflag. From the documentation -

    > Values for oflag are constructed by a bitwise-inclusive OR of flags from the following list, defined in <fcntl.h> O_RDONLY specifies the file is Read Only since we do need to do any editing this is the perfect choice.

  - The return value is stored in a int variable file descriptor which is then used to check for errors. A -1 value stored indicates error while reading and hence an perror is raised.
  - Some variables are initialized for the process of reading and a read syscall is made with the arguments filedescriptor denoting the file which needs to be read, a buffer size which counts which specifies how many bytes from the file should be read and the count denotes the count parameter.
  - The errors are checked if the returned value of read is less than 0 a perror is raised.
  - The csv file is then read line by line in the data matrix
  - Then using strtok() and atoi() tokens are generated and numeric tokens are converted to number from originally being strings. The scores are stored in the marks matrix and sections are stored in the sections array.
  - Finally the close() syscall is used with the filedescriptor to close reading the file since the data is already extracted. Incase it has an error which is when it returns -1 a perror is raised denoting the same.

- stringPrinter(char *str)

  - It is used to print the value written using snprintf into String str.
  - write() syscall is used with the parameter 1, str and length of the string str. The 1 denotes the filedescriptor where it has to be written i.e. standard output in this case. str denotes the string which needs to be sent to stdout and the length tells the syscall how many bytes will be there.
  - Incase of an error a -1 is returned by write which is handled by the if block. Subsequently if errno saved by write is EINTR it means "The call was interrupted by a signal before any data was written" and hence an appropriate perror is raised.

- printforgivensection(char* section, char* sectionTracker[rows-1], int marks[rows-1][6], struct marks_structure *marks_store,int *section_size, int r)

  - 6 Variables are initialized to store the different collective scores of the assignments. A counter is mainted to count how many submissions were there in a given section.
  - The for loop then runs and at any place if the section in the sectionTracker array matches the required section parameter it increments counter and adds the value in the variables.
  - Finally the section size and total marks for individual assignments are stored inside the section_size variable and the marks_store structure respectively.
  - The marks_store is a marks_structure struct which has the following form and hence all assignment scores are stored within it-

```
struct marks_structure
    {
        int total_assignment1;
        int total_assignment2;
        int total_assignment3;
        int total_assignment4;
        int total_assignment5;
        int total_assignment6;
    };
```

- Finally the average is computed and stored in avg variables for respective assignments. The strings are then passed inside snprintf with the required formatting of what needs to be printed which are then sent to the stringPrinter function. snprintf takes the String where the value is to be stored, the size of the buffer and a C string which denotes the format.