Uitility Handbook
Assignment 1: Vote Count Application

This is a utility guide to various utility functions provided to you to complete this assignment.
The open source utilities are present in makeargv.h header file except for two: file_open and read_line, which are provided in binaries.
Some other utilities not associated with makeargv.h are given as executables.
Note: The Usage information below omits the error handling part.

Makeargv.h

1) makeargv : This function splits your string with the specified delimiter.
   Function Header: int makeargv(const char*s, const char *delimiters, char ***argvp)
   Returns: int - number of tokens the string is split into
   Arguments:
   const char*s - String to be split
   const char *delimiters - This is the delimiter you would like to split with
   char ***argvp - Array of output strings

   Sample Usage:
   ```
   if (makeargv(s, " : ", &strings) == 0){
               return 0;
      }
   ```

2) trimwhitespace : This is a utility to trim the string of trailing or beggining white space. You may not need trimming in every case, but it's often needed.

Function Header: char* trimwhitespace(char *str)
Returns: char* - The trimmed string
Arguments:
    const* str - The string to be trimmed

Sample Usage:
```
trimwhitespace(str);
```

3) prepend : This is a utility to prepend a string to a given string.
Function Header: char* prepend(char* s, const char* t)
Returns : char* - The prepended string
Arguments :
char* s - String to be prepended to.
const char* t - String to be prepended to the other string

Assumptions: s has enough space to hold the prepended string. Error will be thrown if not.

Usage:

```
char* str1="Output_";
char* str2=malloc(sizeof(char)*100);
str2="Sub_County_1"
prepend(str2,str1);
```

Result: str2 becomes Output_Sub_County_1

4) file_open - This function is used to open a file.

Function Head: FILE* file_open(char* file_name);
Returns: FILE* - The file pointer
Arguments:
char* file_name - File name of the file to be opened.

Assumptions: The file you would like to open is located in the folder which the program is located in.

Usage:

```
char* filename = "input.txt";
FILE* f = file_open(filename);
```

5) read_line - This function reads characters from the stream stream up to and including a newline character and stores them in the string s, adding a null character to mark the end of the string.

Function Head: char* read_line(char* buffer, FILE* fp);

Returns: char* - The file pointer
Arguments:
char* file_name - File name of the file to be opened.

Assumptions: The file you would like to open is located in the folder which the program is in.

Usage:
char *buf;
..malloc buf to 1024 characters.
buf=read_line(buf,f)

Executables

1) leafcounter

   Arguments expected: InputFile, OutputFile, Number of Candidates, Names of Candidates all separated with space, in this exact same order.

   It opens the InputFile, and parses it, line by line, to count the votes of each of the Candidates that you have specified.

   Output: It outputs a file of the name "OutputFile" argument(2nd input argument).
   The file will contain the name of the candidate and their votes on each new line.
   Usage: ./leafcounter Sub_County_1 Output_Sub_County_1 3 A B C
   Thus considering a file
   Sub_County_1
   A
   A
   A
   B
   B

   The output **Output_Sub_County_1** will look like:
   A 3
   B 2
   C 0

2) aggregate_votes

Arguments expected: Number of Input File, InputFile names, OutputFile name, Number of Candidates, Names of Candidates all separated with space, in this exact same order.

It opens the InputFiles, and parses them, line by line, aggregates the votes of each of the Candidates that you have specified.

Output: It outputs a file of the name "OutputFile name" argument.

The file will contain the name of the candidate and their votes on each new line.

Usage: ./aggregate_votes 3 Output_County_1 Output_County_2 Output_County_3 Output_Region_1 3 A B C

Thus considering an input file

Output_County_1

A 3
B 2
C 1

Output_County_2

A 3
B 2
C 2

Output_County_3

A 3
B 2
C 0

The output **Output_Region_1** will look like:

A 9
B 6
C 3

3) find_winner

Arguments expected: Number of Input File, InputFile names, OutputFile name, Number of Candidates, Names of Candidates all separated with space, in this exact same order.

It opens the InputFiles, and parses them, line by line, aggregates the votes of each of the Candidates that you have specified. Then it compares these values to declare a winner.

Output: It outputs a file of the name "OutputFile name" argument.
The file will contain the name of the candidate and their total votes on each new line.
The last line will have "Winner is: <Name>"

Usage: ./find_winner 3 Output_Region_1 Output_Region_2 Output_Who_Won 3 A B C
Thus considering an input file
Output_Region_1
A 3
B 2
C 1

Output_Region_2
A 3
B 2
C 2

The output **Output_Region_1** will look like:
A 6
B 4
C 3
Winner is: A