File splitter

This assignment will give your practice implementing working with C file input/output, working with files, C/C++ mixing.

Goal: create file splitter/joiner application that allows the user to

- split a file into pieces of a given size – imaging a situation when you need to transfer 1Gb file named "IN" from one computer to another using only 128Mb=134217728 bytes flash drive.
  `./prog.exe -s 134217728 -o chunk_ -i IN`
  takes file "IN" and splits it into chunk_0000, chunk_0001, chunk_0002, ... all chunks except probably the last one have size `134217728` bytes.
- combine several pieces into a single file – in the above example merging all chunks into the original file.
  `./prog.exe -j -o collected -i chunk1 chunk2 chunk3`
  merges files "chunk1", "chunk2", "chunk3" into a file named "collected". The size of "collected" should be exactly the sum of the sizes of "chunk1", "chunk2", and "chunk3".
- command line switches:
  - -j join
  - -s split, followed by the desired chunk size
  - -i input filename(s)
  - -o
    - output filname when used with -j
    - chunk prefix when used with -s

Incomplete interface (doesn't allow C/C++ mixing)

```
#ifndef SPLITTER_H
#define SPLITTER_H
enum {E_BAD_SOURCE=1, E_BAD_DESTINATION, E_NO_MEMORY, E_NO_ACTION,
E_SMALL_SIZE};

int SplitFile(char*  filename,  char* output, size_t size);
int JoinFiles(char** filenames, char* output);
#endif
```

Additional notes:

1. assume binary input and output and use `fread/fwrite`
2. use read buffer size set to the minimum of the desired chunk size and 4K=4096 .
3. buffer should be allocated dynamically using malloc,
4. I/O functions like `fseek, rewind`, etc should not be used.
5. Return codes:
   A) E_BAD_SOURCE – if input file(s) cannot be open
   B) E_BAD_DESTINATION – if output file(s) cannot be open
   C) E_NO_MEMORY – if malloc fails
   D) E_NO_ACTION – used in driver only
   E) E_SMALL_SIZE – currently not used
6. chunk suffix is a 4 digit number padded with 0's, see implementation in `splitter.c`.

7. operating system will expand wildcards for you, so that if you type in the command line
   `./prog.exe -j -o collected -i chunk*`
   and the current folder contains files chunk1, chunk2, chunk3, chunk4, then `argv` array will contain the following:
   ```
   ./prog.exe
   -j
   -o
   collected
   -i
   chunk1
   chunk2
   chunk3
   chunk4
   ```
   see sample implementation file.
8. make sure your code compiles with all 9 targets (see Makefile), there are 3 compilers, and for each compiler your code will be compiled
   A) with C++ compiler (driver.cc and splitter.c)
   B) with C compiler (driver.c and splitter.c)
   C) with C++ compiler (driver.cc) and C compiler (splitter.c), then linked with C++ linker.
9. make sure CodeGuard runs cleanly (target **bcc_cg**)
10. make sure your code compiles with `doxygen` (download Doxygen configuration `Doxyfile` from this assignment folder and save it in the same folder with your code).

Usefull tools: (*nix,Cygwin – but there are Windows equivalents)

– "od" -- octal dump (Cygwin's coreutils package), which may help you to examine the produced file. Usage
```
od -x filename
```
output
```
0000000 6923 6e66 6564 2066 4946 454c 414e 454d
0000020 485f 0a0d
0000024
```
where the left column contains addresses, all other columns are ASCII codes in hexadecimal format corresponding to characters in the file, note that "6923" represents 2 characters – 23 and 69 (**in this order!!!**) which are '#' and 'i'. Also note that the file was in Windows text format, since we can clearly see "0a0d " in the end, which is Windows CRLF (carriage return "0d ", line feed "0a") line terminator.

– "diff" – see a link on class web-page for more information.
Very basic usage:
```
diff file1 file2
```
output will look like
```
2c2
< a
---
> b
```
showing that there is a difference in line 2, which is 'a' in the first file and 'b' in the second. (I used 2 files with the same first line and second line in both files was a single character)

```
diff file1 file2 –strip-trailing-cr
```
"--strip-trailing-cr" flag makes "diff" to NOT pay attention to the newline style (do not use this flag for this assignment, newlines have to be preserved)

If there is a single line of output from “diff” - there is(are) differences, and you should figure out were exactly they are. Use

```
diff file1 file2 -y
```

“-y” flag makes “diff” to display the output in two columns, marking lines that differ with a bar '|'.

```
1                               1
a                       |       b
```

Hint: 2 columns will not fit into a terminal, so you should either resize the terminal (click icon “C:\”, then Properties->Layout->Window Size), or redirect output into a file (add “ > difference.txt” in the end of diff line, and then open file “difference.txt” in any editor)

Submission:

submit electronically 3 files:

```
splitter.c
splitter.h
index.chm
```