```
/* minicat.c
 * Last changed: Sept. 18, 2018
 * Author: Min Joon So
 */

#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>

/* Minicat
 Description: Concatenates and copies files

 Usage:
        minicat [-b ###] [-o outfile] infile1 [...infile2...]
        minicat [-b ###] [-o outfile]
*/

// function to read an infile until the EOF and writes to 'outfile'
int concat(int descriptor_in, int descriptor_out, long buffersize, char* buffer,
char* output_file, char* input_file){
        int bytes_written = 0;
        int bytes_read = read(descriptor_in, buffer, buffersize);

        while(bytes_read > 0){
                //printf("looping\n");
                bytes_written = write(descriptor_out, buffer, bytes_read);
                if(bytes_written < 0){
                        fprintf(stderr,"ERROR (%s): Failed to write to output file
%s\n", strerror(errno), output_file);
                        exit(-1);
                }
                bytes_read = read(descriptor_in, buffer, buffersize);
        }
        if(bytes_read < 0){
                fprintf(stderr,"ERROR (%s): Failed to read an input file %s\n",
strerror(errno), input_file);
                exit(-1);
        }
}

int main(int argc, char **argv){
        int c;
        int index;
        int descriptor_out = STDOUT_FILENO;
```

```
        int descriptor_in = -1;
        long buffersize = 1024;                         //Default to 1024
        char* output_file = 0;

        // check for valid option flags and input arguments
        while ((c = getopt(argc, argv, "b:o:"))!= -1){
                switch(c){
                case 'b':
                        if((buffersize = atoi(optarg)) < 1){
                                fprintf(stderr, "ERROR: Invalid argument: buffer
size must be greater than 0\n");
                                exit(-1);
                        }
                        break;
                case 'o':
                        output_file = optarg;
                        descriptor_out = open(optarg, O_RDWR|O_CREAT|O_TRUNC,0666);
                        if(descriptor_out < 0){
                                fprintf(stderr,"ERROR (%s): Failed to OPEN output
file %s\n", strerror(errno), output_file);
                                exit(-1);
                        }
                        break;
                case '?':
                        if(optopt=='b'){
                                fprintf(stderr,"ERROR: -b argument missing\n");
                                exit(-1);
                        }else if(optopt=='o'){
                                fprintf(stderr,"ERROR: -o argument missing\n");
                                exit(-1);
                        }
                        break;
                default:
                        fprintf(stderr, "ERROR: Incorrect input\n");
                        exit(-1);
                }
        }

        //initialize buffer
        char *buffer = malloc(buffersize);

        // loop through infiles
        for(index = optind; index < argc; index++){
                if(!strcmp("-", argv[index])){
                        descriptor_in = STDIN_FILENO;
                }else{
                        descriptor_in = open(argv[index], O_RDONLY);
                }
```

```
                if(descriptor_in < 0){
                        fprintf(stderr, "ERROR: Failed to OPEN a input file
%s\n",argv[index]);
                        exit(-1);
                }
                concat(descriptor_in, descriptor_out, buffersize, buffer,
output_file, argv[index]);
        }

        // if not input files
        if(descriptor_in == -1){
                descriptor_in = STDIN_FILENO;
                concat(descriptor_in, descriptor_out, buffersize, buffer,
output_file, argv[index]);
        }

        if(descriptor_out != STDOUT_FILENO && close(descriptor_out) < 0 ){
                fprintf(stderr,"ERROR (%s): Failed to CLOSE output file %s\n",
strerror(errno), output_file);
                exit(-1);
        }
        free(buffer);
        return(EXIT_SUCCESS);
}
```