Design Document: Dog

Julia Sales
CruzID: jesales

1) Goals

The goal of this program is to perform functionality similar to that of the cat function in terminal. Dog will read the files entered and print out the contents of each file. Dog needs to be able to handle reading and out putting the contents of multiple files to the command line.

Dog needs to be able to handle dashes (-), process files one at a time, needs to process binary, and have error messages.

2) Design

Part 1: Handling and Reading the arguments
Part 2: Process each argument and print results

2.1 Handling and Reading the arguments

In this part of the program, we initialize our variables. We define our argument count, and we set up an array to hold the arguments.

Reminder: import needed libraries for allowed library functions

Input: Argument count: arg_count
Input: Array of arguments: arg
Output: contents of the files: std_out

```
1-  int main (int arg_count, char *arg[ ]) *from stack overflow*
2-  arg_count <- cmd line
3-  char +arg[ ] <- datatype(char) args[arg_count]
4-  int <- i, j, in;
5-  char <- "-"
6-  char <- buffer
7-  if arg_count = 1 then
8-      |  read (stdin, buffer, size of buffer)
9-      |  while (not eof)
10-         |  write (stdout, buffer, size of buffer)
11-         |  read (stdin, buffer, size of buffer)
12-  else
13-     |  for(i=1; i<=arg_count; i++)
14-         |  if arg_count[1] = " - " <-char
15-             |  read (stdin, buffer, size of buffer)
```

```
16-             | while (not eof)
17-                 | write (stdout, buffer, size of buffer)
18-                 | read (stdin, buffer, size of buffer)
19-         | else
20-             | Process each argument (part 2.2)
```

2.2 Process each argument and print results

In this part of the program, we read each file one by one and take the contents of each file and print them to stdout. This part will be inside the for loop in 2.1. This logic should also be able to process .txt and .bin files.

```
1-      | open file
2-      | initialize buffer
3-      | if files does not exist
4-          | print error message
5-      | else
6-          | read the file
7-          | if file cannot be read
8-              | print error message
9-          | write out contents of file
10-             | while file is bigger than buffer then
11-                 | write
12-                 | reread file
10-   | close files
```