implementCommand will take the array of the command line arguments and the length of the array. It would not return anything. This function will create a child process and will use the child process to implement the command while the parent process will wait the child process until it finishes if the command does not have & at the end. If the command has & at the end, the parent would not wait the child to finish.

Cstring will receive C++ string and C string and it will convert the C++ string to C string. The function would not return anything. In addition, the function pass by reference the index of the of the last argument in the command.

history is a function that will take a list of all the commands the user inserted and will return a list of the 10 most recent commands.

On this project, two vectors are used. One for storing the 10 most recent commands and another for storing all commands the user inserted. Storing the 10 most recent commands is divided into two parts. The first one is when the user inserted less than 10 commands. In this condition, all the commands that are the user inserted are consider the 10 most recent commands. The second condition is when the user inserted more than 10 commands. In this condition, we start with the most recent command and we add it to the 10 most recent commands vector. We keep doing that until we reach the 11th recent element then we stop adding more commands.

Test plan:

- ls
    - will list the files in the current directory.
- history
    - will list the last 10 commands implemented.
- !!
    - will implement the last command and will echo the command that was implemented. The command will be added to the history list.
- !1
    - will implement the last command and will echo that command.
- !2
  will implement the second last command and will echo that command. The command will be added to the history list. if history list is
    1 ls
    2 pwd
    3 open .
    4 date

after implementing !2, the list will be as following

1 pwd
2 ls
3 pwd
4 open .
5 date

- !11
  - will cause an error message and will ask for the next command.
- exit
  - will terminate the shell.
- open <filename> &
  - will open the file named filename using another process then will wait 2 seconds before the parent process resume receiving the next command.
- cd <subdirectory>
  - **would not** be implemented.
- cal
  - will show the calendar.

Answers for the four questions:
1- Converting C++ strings to C strings since execvp do not accept C++ strings.
2- I found creating a new process and dealing with two processes, parent and child, is the least difficult about process manipulation.
3- Can not think of any.
4- Did not expect I could build a simple shell using a process manipulation.

Bugs:

- Sometimes it requires inserting exit command couple of times for the program to exit. I can not understand why this is happening!