In the main function, the code first initialize last executed command to null, then it enters a while loop for Shell. First it initializes variables that are needed for parsing tokens. Then it calls printPrompt function to print a prompt for user. This function gets the username, hostname and current working directory and writes a prompt in requested form. Then the code gets input line from the user and writes it in cmdLine variable. Then it checks if the user just hit the enter and if so it continues to while loop. Then it initializes background variable for background processes and some copies of our cmdline to use later. Then it takes first token and checks if its empty, meaning user just entered space, and continue while loop if so. Then it checks whether the token is exit if it is, the code first free allocated memories then exits. Then checks if the first token is alias, if it is, code parses the line in alias format and creates new alias by calling addAlias function. This function gets a name and command, both strings, and checks whether a file is created for storing alias commands, if there are no files created, it creates a file named AliasFile and writes the alias there, if there is a file already, it directly writes the alias name and command. The last executed command become alias if it executed correctly. Then the while loop continues. If the token was not exit or alias, code checks if there is an alias defined for the token by calling findAlias function. This function checks if there is a file named AliasFile exists and searches the token in the file, returns the command if found. Otherwise the function returns null. If findAlias function do not return null, the command in alias is added to the cmdLine and continue parsing this new command line. Then the parsing for first command line completes with a while loop and this gives the number of tokens in line (argc). Memory is allocated for argv (tokens) and another while loop starts parsing the copy of cmdline and assigning tokens in argv. Then code checks “&” sign and sets background flag. Then if the command is bello, a function called displayUserInfo called with parameter last executed command, and this function prints out the info that is required (except the process count, i could not handle it ☹). If the command is not bello, executeCommand function is called with argv , background and argc parameters. In execute command function, the code does the initializations and checks for redirection operations, if there are >, >> or >>> in command, it sets the correct flags and outputfile variable, and removes the output name and redirection operation sign from argv. Then the command handles path by using “PATH” and parsing accordingly. It then checks whether the path is accesible and sets the flag. Then sets the succes\_executable\_path vaariable if it is accesible, and prints an error message "Command not found” if it isnt. Then it creates a pipet o use later when >>> operation being handled. Theb it calls fork and if the process is not in background, parent process waits fort he child to return. Then in child process,(if pid = 0, meaning we are in child process) checks the output file and flags and sets the right flags for file. If the command includes >>>, the code uses pipe and redirects stdout to the pipe. Else it creates the file accordingly. Then it calls execv with path and argv and executes the command. Then if invert flag is set, parent process read the result from pipe, inverts it and write it to the file. Then it returns to the main func and free the mem allocations.