GEBZE TECHNICAL UNIVERSITY

CSE 344
SYSTEM PROGRAMMING
Homework #1 Report

OGUZ MUTLU
1801042624

1.Problem Definition

In this assignment student management system utilizing system call for interaction with the operating system was designed. Student management system should be able to operate and manipulate some data about student name, surname and grade. Management system should be able to;

-Adding new students to system

-Displaying students

-Searching students according to name

-Sorts with respect to name or grades

The system will store student information in a text file, Each line of the file represent student and students' grade, (name surname,grade). Student name and grade data will separated with "," (comma). Students data will separated with "\n" (new line) character. System calls like open, read, write, close, lseek will be used for file manipulation.

The system should handle potential errors gracefully, including;

- system call errors

- argument or user input errors

- file errors

- memory errors

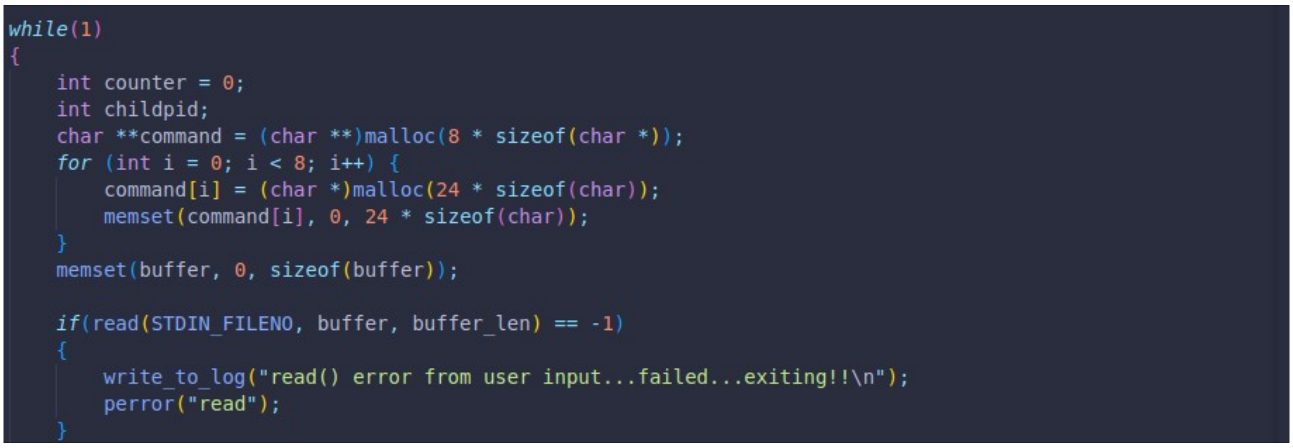
In summary system is considered successful if it can manage student data addition, search, sort, display with completely system calls and handle potential errors appropriately.

2.Implementation

The architecture considered for this assignment will be as follows.

-Since the incorrect return of system calls such as fork should be written into the log file, the log file was kept specific to the parent, therefore it was not run with a fork. The log file is created again every time the program is started.

-Since it will be done with a system call, it must work like a terminal while the program is running. Therefore, with the read() system call, input from the user is expected in an endless loop and recorded in the buffer. Since input will be received from the user in the parent process, it was done without using a fork. Appropriate error has been handled.

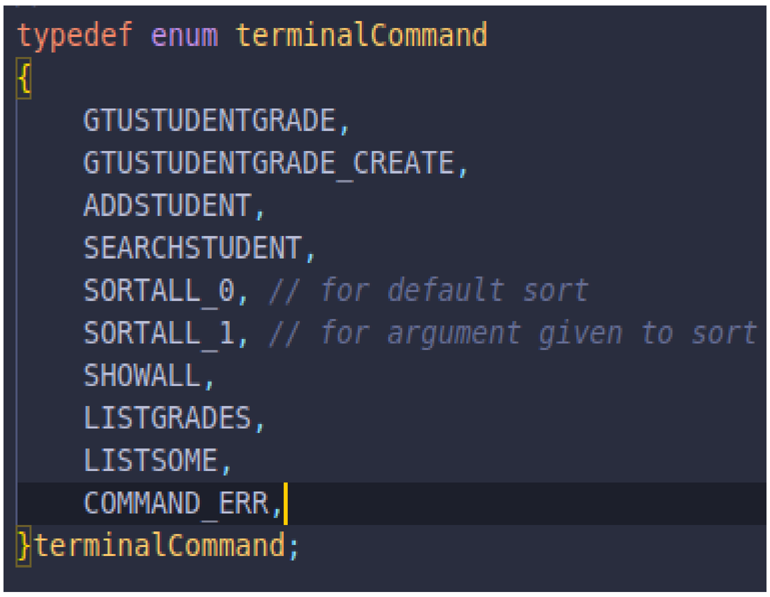


```c
while(1)
{
    int counter = 0;
    int childpid;
    char **command = (char **)malloc(8 * sizeof(char *));
    for (int i = 0; i < 8; i++) {
        command[i] = (char *)malloc(24 * sizeof(char));
        memset(command[i], 0, 24 * sizeof(char));
    }
    memset(buffer, 0, sizeof(buffer));

    if(read(STDIN_FILENO, buffer, buffer_len) == -1)
    {
        write_to_log("read() error from user input...failed...exiting!!\n");
        perror("read");
    }
```

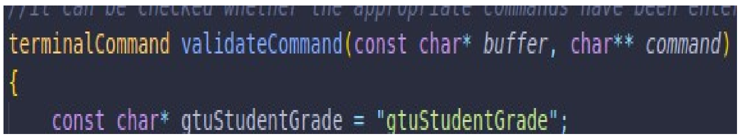
Figure 1. user input read() syscall

-After receiving the command, it must be checked whether it complies with the requirements of the program. For this reason, an enum structure was used. With this enum structure, the integer value specific to the command is returned or the integer value is returned upon command error.

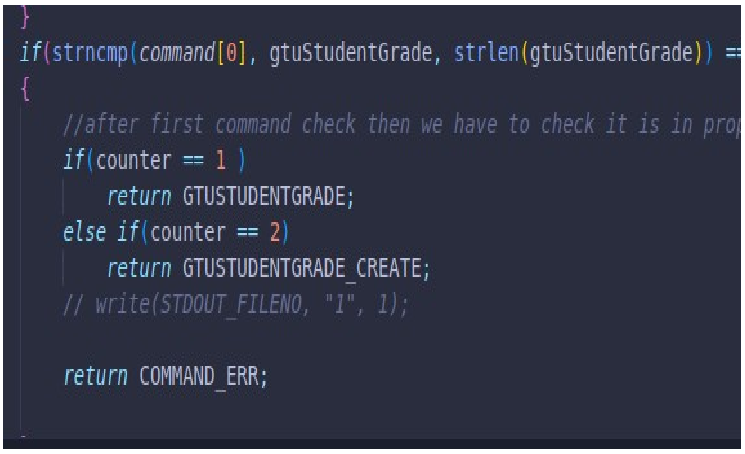
Thanks to the validateCommand function, it can be checked whether the appropriate commands have been entered or the appropriate number of arguments have been entered.



```c
typedef enum terminalCommand
{
    GTUSTUDENTGRADE,
    GTUSTUDENTGRADE_CREATE,
    ADDSTUDENT,
    SEARCHSTUDENT,
    SORTALL_0, // for default sort
    SORTALL_1, // for argument given to sort
    SHOWALL,
    LISTGRADES,
    LISTSOME,
    COMMAND_ERR,
}terminalCommand;
```



```c
//it can be checked whether the appropriate commands have been enter
terminalCommand validateCommand(const char* buffer, char** command)
{
    const char* gtuStudentGrade = "gtuStudentGrade";
```



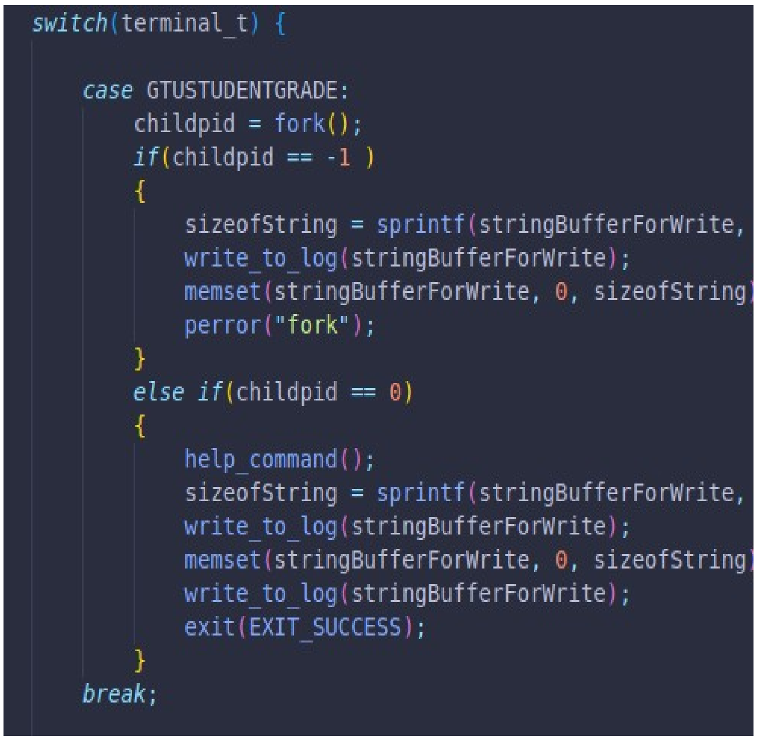
```c
}
if(strncmp(command[0], gtuStudentGrade, strlen(gtuStudentGrade)) ==
{
    //after first command check then we have to check it is in prop
    if(counter == 1 )
        return GTUSTUDENTGRADE;
    else if(counter == 2)
        return GTUSTUDENTGRADE_CREATE;
    // write(STDOUT_FILENO, "1", 1);

    return COMMAND_ERR;
```

Figure 2. enum declaration-validateCommand function example

NOTE: All parsing operations were done with the help of string.h library. Parsing operations are performed according to the **space** character, **new line** character or **comma** character, depending on the requirements.

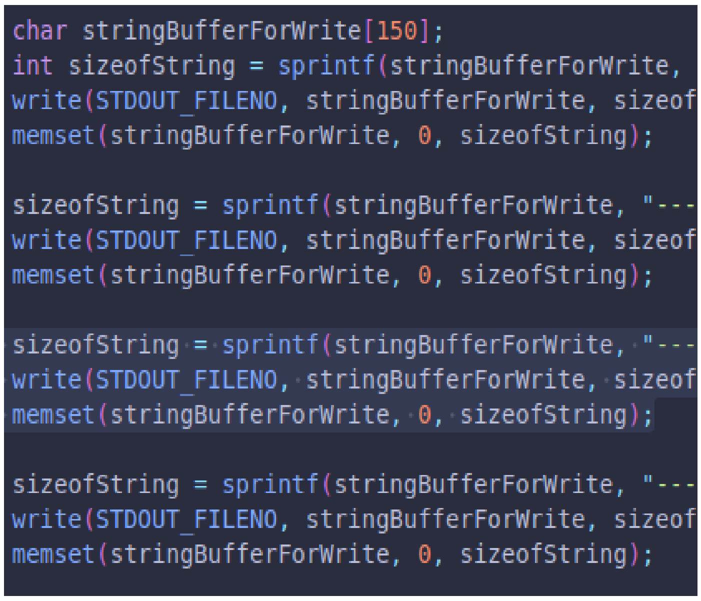
- If the parsed command returns *GTUSTUDENTGRADE* from the validateCommand function, this structure is entered in the switch case structure and the fork is run. For childpid == 0, constant strings in the helpCommand function are written to the terminal screen (*STDOUT_FILENO*) with the write system call.The parent process starts waiting for child process. After the operation logs are written to the log.txt file the child exits with SUCCESS.

-If the parsed command returns the GTUSTUDENTGRADE_CREATE value from the validateCommand function, this structure is entered into the switch case structure and the fork is run. In this switch case structure, a file is created according to the argument name entered in the child process. Each time this command is entered, the file is created if it does not exist, and truncated if it exists.After the operation logs are written to the log.txt file the child exits with SUCCESS.

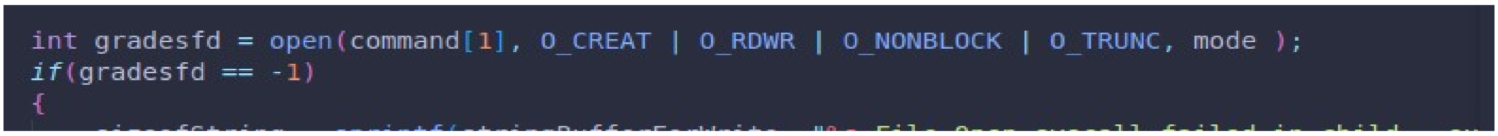


```
switch(terminal_t) {

    case GTUSTUDENTGRADE:
        childpid = fork();
        if(childpid == -1 )
        {
            sizeofString = sprintf(stringBufferForWrite,
            write_to_log(stringBufferForWrite);
            memset(stringBufferForWrite, 0, sizeofString)
            perror("fork");
        }
        else if(childpid == 0)
        {
            help_command();
            sizeofString = sprintf(stringBufferForWrite,
            write_to_log(stringBufferForWrite);
            memset(stringBufferForWrite, 0, sizeofString)
            write_to_log(stringBufferForWrite);
            exit(EXIT_SUCCESS);
        }
    break;
```

```
void help_command()
{
```



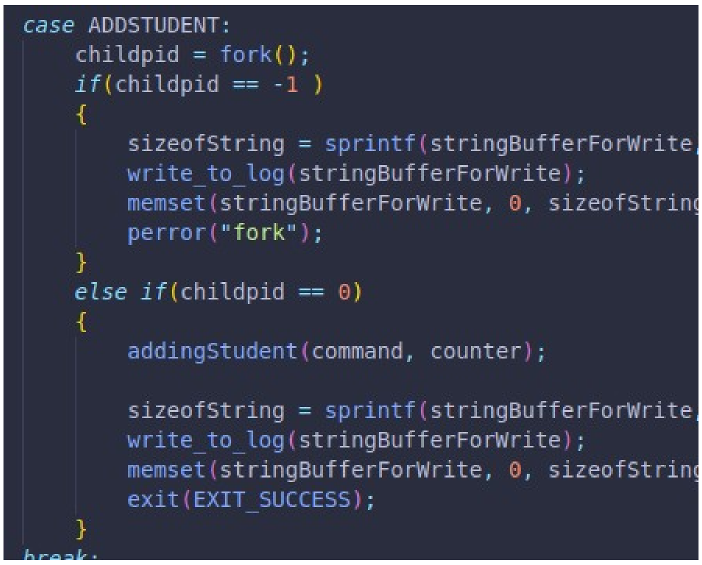
```
char stringBufferForWrite[150];
int sizeofString = sprintf(stringBufferForWrite,
write(STDOUT_FILENO, stringBufferForWrite, sizeof
memset(stringBufferForWrite, 0, sizeofString);

sizeofString = sprintf(stringBufferForWrite, "---
write(STDOUT_FILENO, stringBufferForWrite, sizeof
memset(stringBufferForWrite, 0, sizeofString);

sizeofString = sprintf(stringBufferForWrite, "---
write(STDOUT_FILENO, stringBufferForWrite, sizeof
memset(stringBufferForWrite, 0, sizeofString);

sizeofString = sprintf(stringBufferForWrite, "---
write(STDOUT_FILENO, stringBufferForWrite, sizeof
memset(stringBufferForWrite, 0, sizeofString);
```



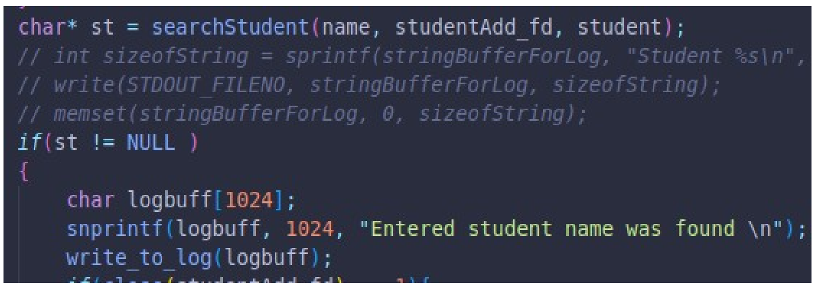
```
int gradesfd = open(command[1], O_CREAT | O_RDWR | O_NONBLOCK | O_TRUNC, mode );
if(gradesfd == -1)
{
```

Figure 3: switchcase structure, fork and helpCommand function, openflags and modes

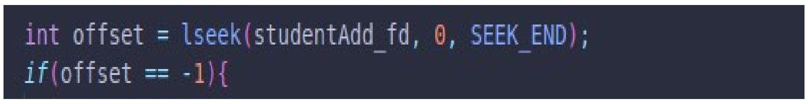
-If the parsed command returns the ADDSTUDENT value from the validateCommand function, this structure is entered into the switch case structure and the fork is run. In this switch case structure, the argument entered in the child process is parsed according to its names and its structure is written to the file as "name, surname, grade\n". If the student given in the argument is in the file, no change is made and the user is informed that this student exists on the terminal screen and log file.



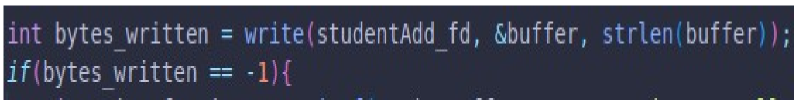
```
case ADDSTUDENT:
    childpid = fork();
    if(childpid == -1 )
    {
        sizeofString = sprintf(stringBufferForWrite,
        write_to_log(stringBufferForWrite);
        memset(stringBufferForWrite, 0, sizeofString
        perror("fork");
    }
    else if(childpid == 0)
    {
        addingStudent(command, counter);

        sizeofString = sprintf(stringBufferForWrite,
        write_to_log(stringBufferForWrite);
        memset(stringBufferForWrite, 0, sizeofString
        exit(EXIT_SUCCESS);
    }
```



```
char* st = searchStudent(name, studentAdd_fd, student);
// int sizeofString = sprintf(stringBufferForLog, "Student %s\n",
// write(STDOUT_FILENO, stringBufferForLog, sizeofString);
// memset(stringBufferForLog, 0, sizeofString);
if(st != NULL )
{
    char logbuff[1024];
    snprintf(logbuff, 1024, "Entered student name was found \n");
    write_to_log(logbuff);
```

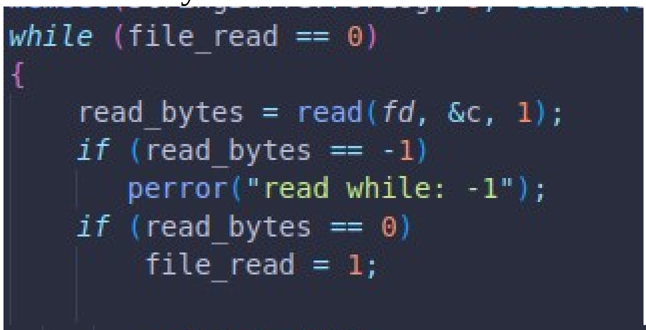


```
int offset = lseek(studentAdd_fd, 0, SEEK_END);
if(offset == -1){
```

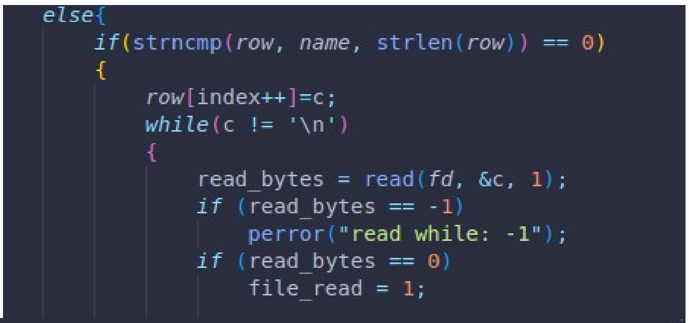


```
int bytes_written = write(studentAdd_fd, &buffer, strlen(buffer));
if(bytes_written == -1){
```

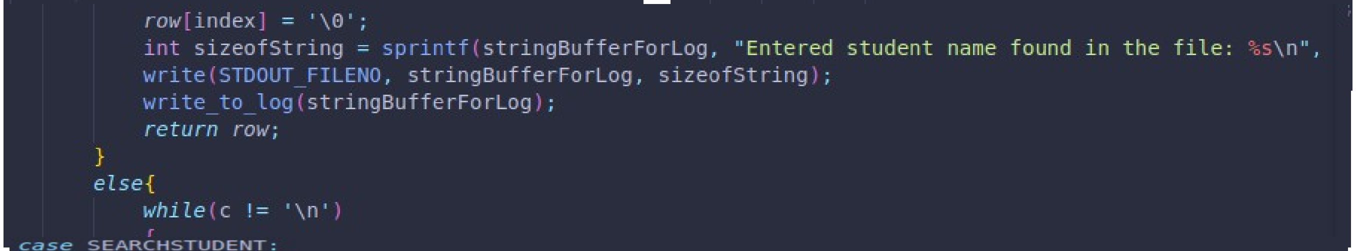
-If the parsed command returns the SEARCHSTUDENT value from the validateCommand function, this structure is entered into the switch case structure and the fork is run. In this switch case structure, the arguments entered into the child process are parsed according to their names. After the file to be read starts to be read with system calls read(), it is read up to the "," character and compared with the argument with strncmp. If the result returns null, the entered student is not found in the file, if not, it returns the student's grade and name. The child then terminates itself successfully.



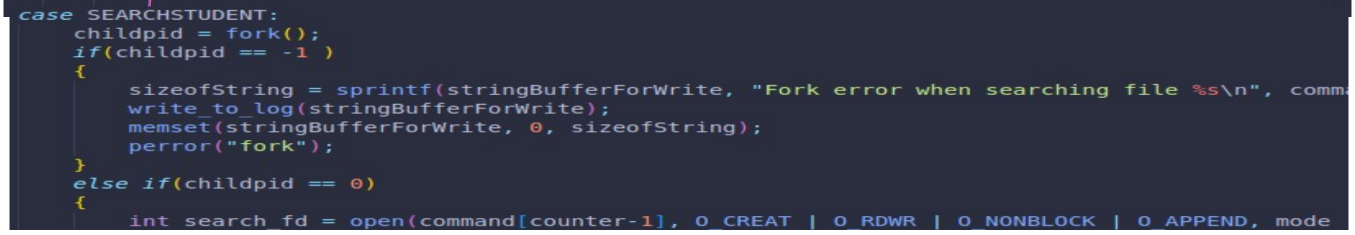
```
while (file_read == 0)
{
    read_bytes = read(fd, &c, 1);
    if (read_bytes == -1)
        perror("read while: -1");
    if (read_bytes == 0)
        file_read = 1;
```



```
else{
    if(strncmp(row, name, strlen(row)) == 0)
    {
        row[index++]=c;
        while(c != '\n')
        {
            read_bytes = read(fd, &c, 1);
            if (read_bytes == -1)
                perror("read while: -1");
            if (read_bytes == 0)
                file_read = 1;
```

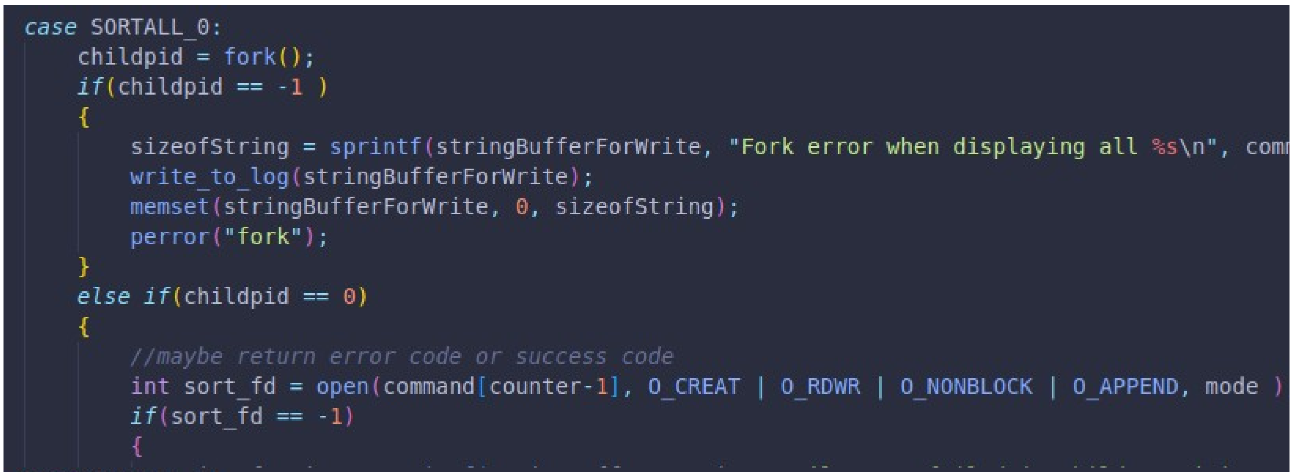


```
            row[index] = '\0';
            int sizeofString = sprintf(stringBufferForLog, "Entered student name found in the file: %s\n",
            write(STDOUT_FILENO, stringBufferForLog, sizeofString);
            write_to_log(stringBufferForLog);
            return row;
        }
        else{
            while(c != '\n')
```

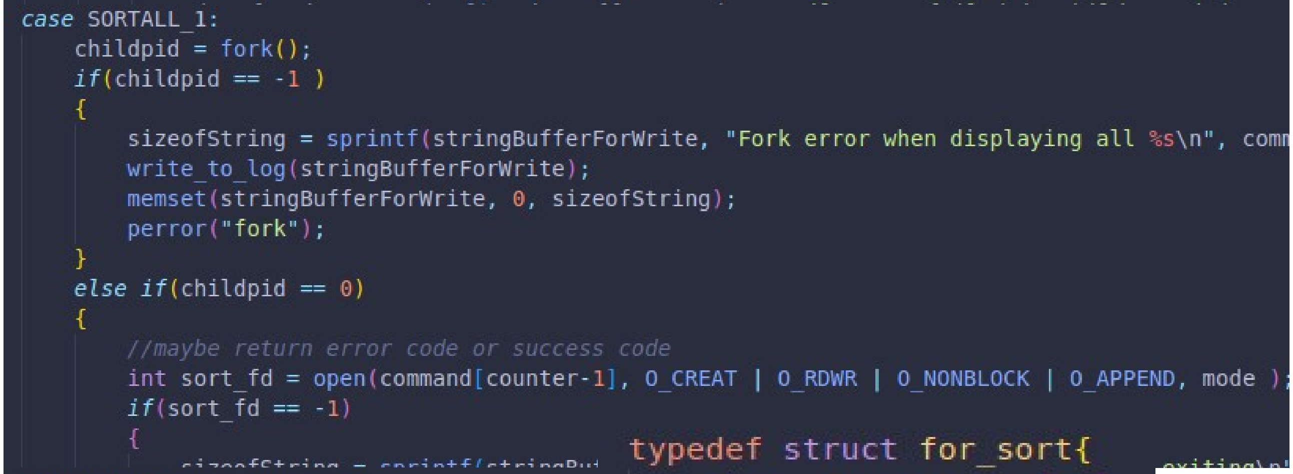


```
case SEARCHSTUDENT:
    childpid = fork();
    if(childpid == -1 )
    {
        sizeofString = sprintf(stringBufferForWrite, "Fork error when searching file %s\n", comm
        write_to_log(stringBufferForWrite);
        memset(stringBufferForWrite, 0, sizeofString);
        perror("fork");
    }
    else if(childpid == 0)
    {
        int search_fd = open(command[counter-1], O_CREAT | O_RDWR | O_NONBLOCK | O_APPEND, mode
```

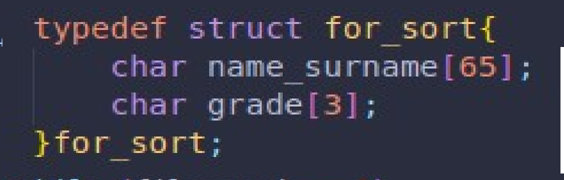
-If the parsed command returns the SORTALL_0 value from the validateCommand function, this structure is entered into the switch case structure and the fork is run. In this switch case structure, the file name argument entered into the subprocess is parsed. _0 is used to indicate the default operation. If the user does not enter any parameters, ascending name order is performed as the default process. Qsort was used for this child process. Each student taken from the file was kept in the struct data structure. Name and surname are read up to the comma, and grade string is read from the comma to the new line character. All students are kept by creating a struct array dynamically.The SORTALL_1 parameter performs the same operations, the only difference from _0 is that it accepts one of the parameters such as "n,g,d,c" when receiving the command from the user. It decides how to rank according to these parameters. Is it increasing or decreasing, increasing according to the name or decreasing according to the note, etc...The child then terminates itself successfully.



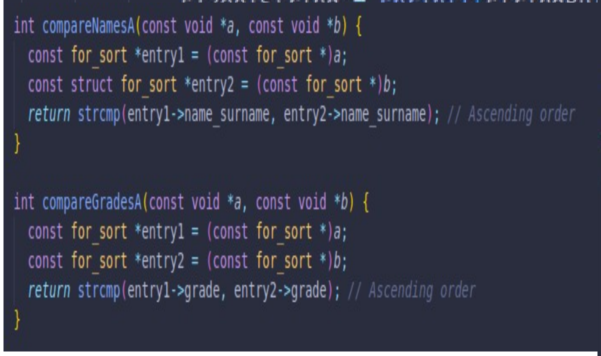
```
case SORTALL_0:
    childpid = fork();
    if(childpid == -1 )
    {
        sizeofString = sprintf(stringBufferForWrite, "Fork error when displaying all %s\n", comm
        write_to_log(stringBufferForWrite);
        memset(stringBufferForWrite, 0, sizeofString);
        perror("fork");
    }
    else if(childpid == 0)
    {
        //maybe return error code or success code
        int sort_fd = open(command[counter-1], O_CREAT | O_RDWR | O_NONBLOCK | O_APPEND, mode )
        if(sort_fd == -1)
        {
```



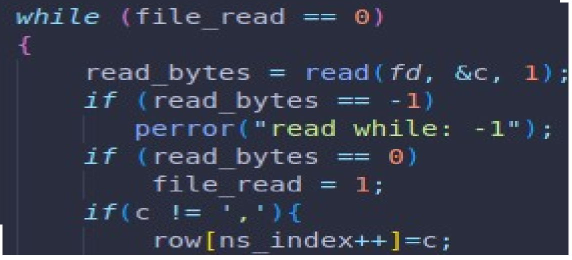
```
case SORTALL_1:
    childpid = fork();
    if(childpid == -1 )
    {
        sizeofString = sprintf(stringBufferForWrite, "Fork error when displaying all %s\n", comm
        write_to_log(stringBufferForWrite);
        memset(stringBufferForWrite, 0, sizeofString);
        perror("fork");
    }
    else if(childpid == 0)
    {
        //maybe return error code or success code
        int sort_fd = open(command[counter-1], O_CREAT | O_RDWR | O_NONBLOCK | O_APPEND, mode );
        if(sort_fd == -1)
        {
```



```
typedef struct for_sort{
    char name_surname[65];
    char grade[3];
}for_sort;
```

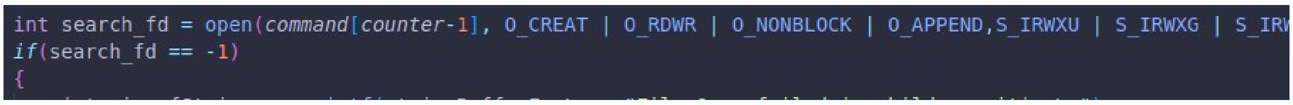


```
int compareNamesA(const void *a, const void *b) {
  const for_sort *entry1 = (const for_sort *)a;
  const struct for_sort *entry2 = (const for_sort *)b;
  return strcmp(entry1->name_surname, entry2->name_surname); // Ascending order
}

int compareGradesA(const void *a, const void *b) {
  const for_sort *entry1 = (const for_sort *)a;
  const for_sort *entry2 = (const for_sort *)b;
  return strcmp(entry1->grade, entry2->grade); // Ascending order
}
```

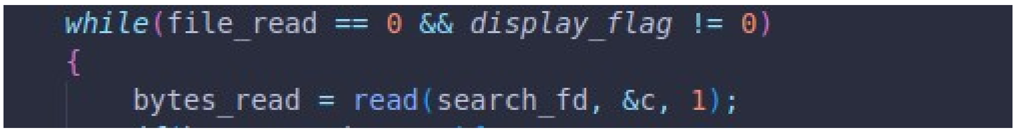


```
while (file_read == 0)
{
    read_bytes = read(fd, &c, 1);
    if (read_bytes == -1)
       perror("read while: -1");
    if (read_bytes == 0)
        file_read = 1;
    if(c != ','){
        row[ns_index++]=c;
```

-If the parsed command returns SHOWALL and LISTGRADES values from the validateCommand function, this structure is entered into the switch case structure and the fork is run. In this switch case structure, the file name, which is the argument entered into the child process, is parsed, then 1 flag value is sent to the same function for SHOWALL and 2 flag values for LISTGRADES. With the read() system call, the file is read and if 1 flag value is received, it is transferred to the end of the file, if 2 flag values are received, 5 flag values are sent. It reads \n characters and writes the results to the terminal.The child then terminates itself successfully.

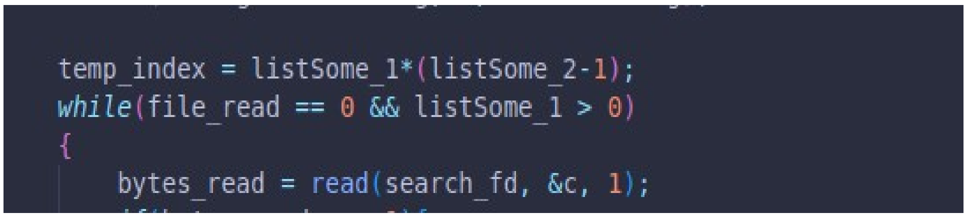


```
int search_fd = open(command[counter-1], O_CREAT | O_RDWR | O_NONBLOCK | O_APPEND,S_IRWXU | S_IRWXG | S_IRW
if(search_fd == -1)
{
```

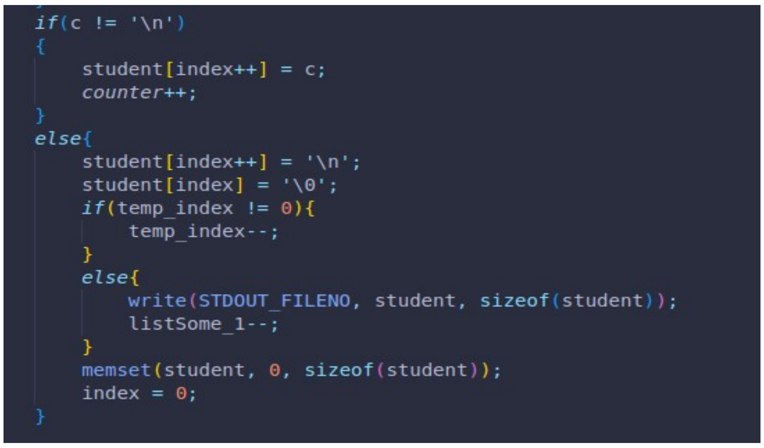


```
    while(file_read == 0 && display_flag != 0)
    {
        bytes_read = read(search_fd, &c, 1);
```

-If the parsed command returns the LISTSOME value from the validateCommand function, this structure is entered into the switch case structure and the fork is run. In this switch case structure, the argument entered to the child process is parsed according to the "count, pagenum, filename" structure and "temp_index = listSome_1*(listSome_2-1);" By doing this, the number of students on the requested page number is written and the child process returns successfully.



```
    temp_index = listSome_1*(listSome_2-1);
    while(file_read == 0 && listSome_1 > 0)
    {
        bytes_read = read(search_fd, &c, 1);
```

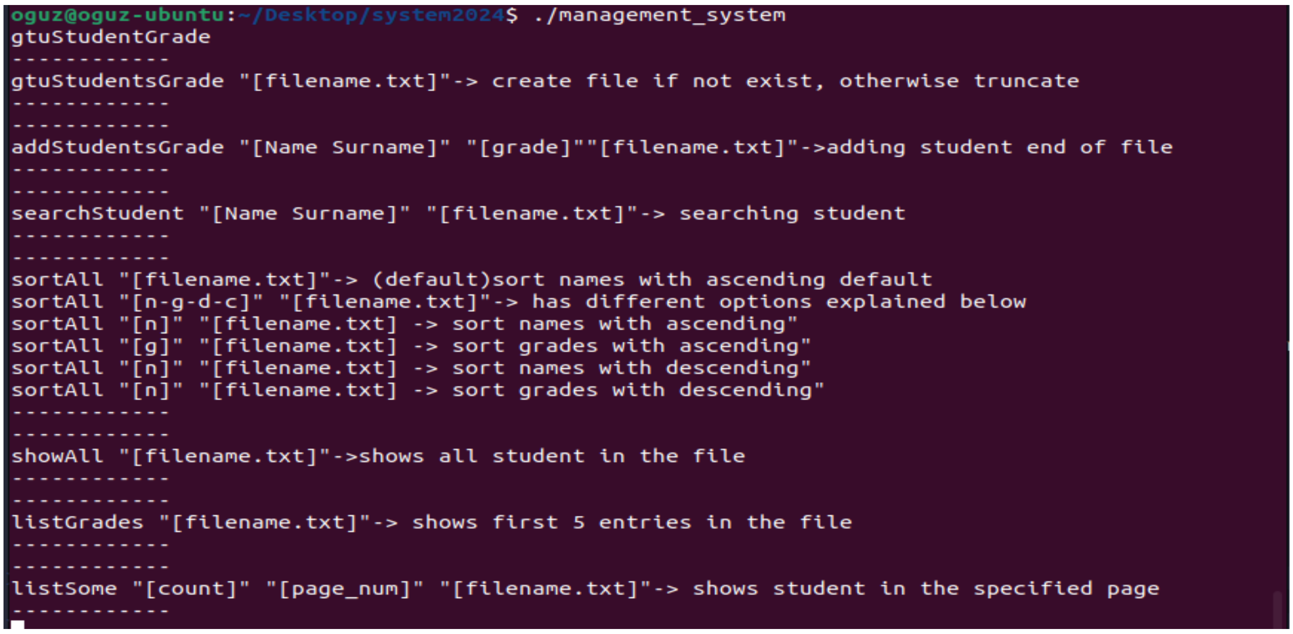


```
if(c != '\n')
{
    student[index++] = c;
    counter++;
}
else{
    student[index++] = '\n';
    student[index] = '\0';
    if(temp_index != 0){
        temp_index--;
    }
    else{
        write(STDOUT_FILENO, student, sizeof(student));
        listSome_1--;
    }
    memset(student, 0, sizeof(student));
    index = 0;
}
```

3.Results and

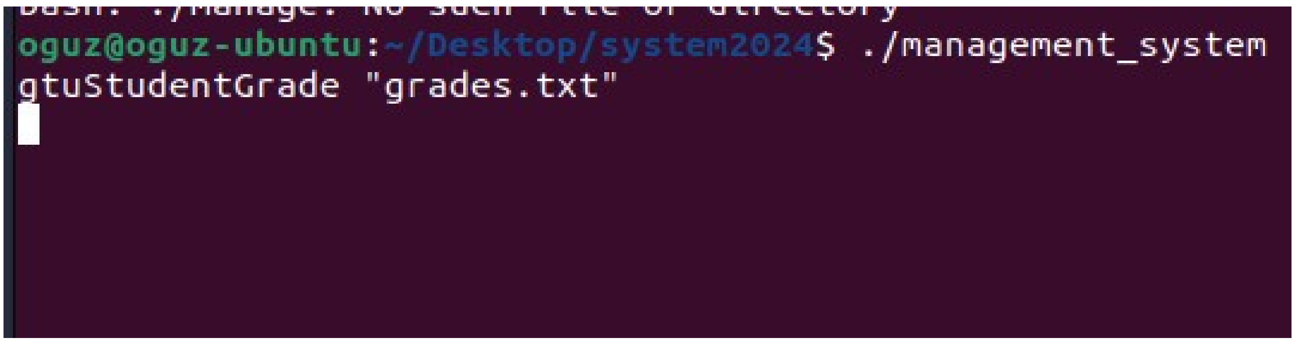
Test Cases

>>gtuStudentGrade with no argument

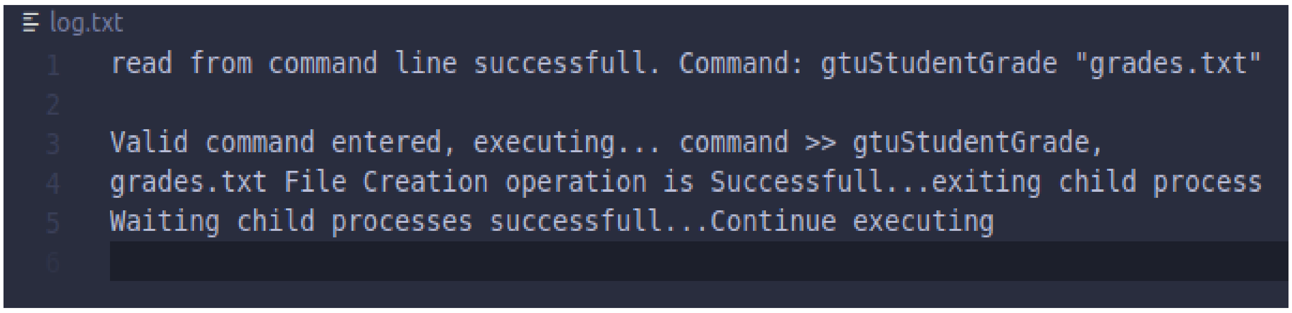


```
oguz@oguz-ubuntu:~/Desktop/system2024$ ./management_system
gtuStudentGrade
------------
gtuStudentsGrade "[filename.txt]"-> create file if not exist, otherwise truncate
------------
------------
addStudentsGrade "[Name Surname]" "[grade]""[filename.txt]"->adding student end of file
------------
------------
searchStudent "[Name Surname]" "[filename.txt]"-> searching student
------------
------------
sortAll "[filename.txt]"-> (default)sort names with ascending default
sortAll "[n-g-d-c]" "[filename.txt]"-> has different options explained below
sortAll "[n]" "[filename.txt] -> sort names with ascending"
sortAll "[g]" "[filename.txt] -> sort grades with ascending"
sortAll "[n]" "[filename.txt] -> sort names with descending"
sortAll "[n]" "[filename.txt] -> sort grades with descending"
------------
------------
showAll "[filename.txt]"->shows all student in the file
------------
------------
listGrades "[filename.txt]"-> shows first 5 entries in the file
------------
------------
listSome "[count]" "[page_num]" "[filename.txt]"-> shows student in the specified page
------------
```

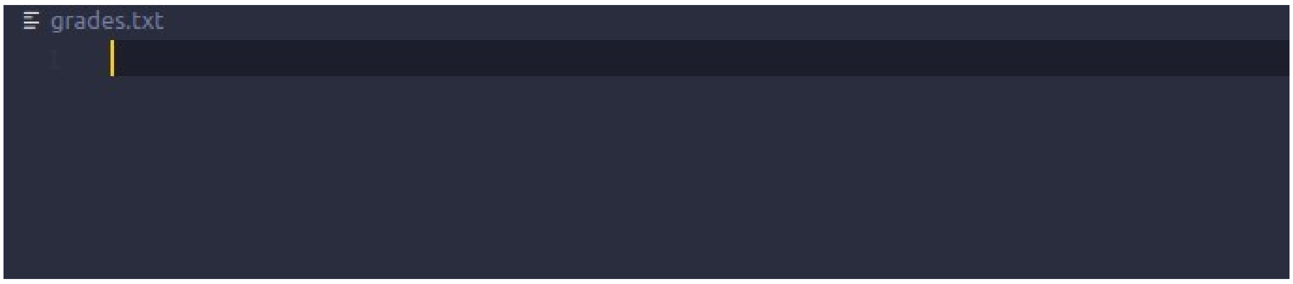
>>gtuStudentGrade “grades.txt”



```
bash: ./manage: No such file or directory
oguz@oguz-ubuntu:~/Desktop/system2024$ ./management_system
gtuStudentGrade "grades.txt"
```

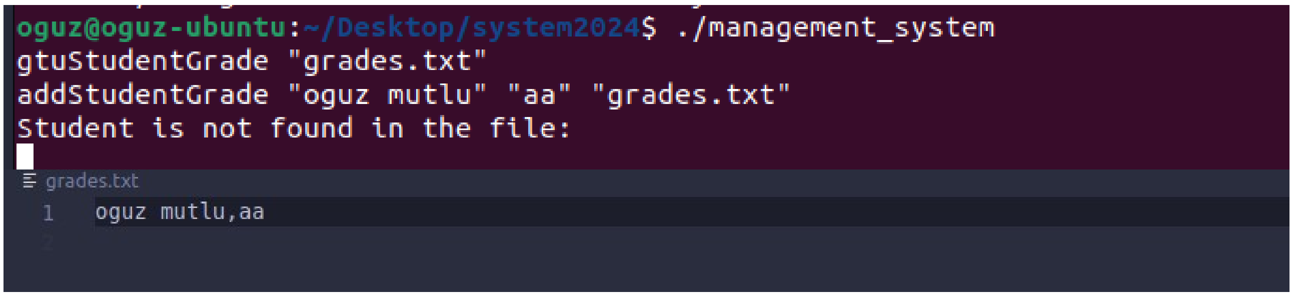


```
log.txt
read from command line successfull. Command: gtuStudentGrade "grades.txt"

Valid command entered, executing... command >> gtuStudentGrade,
grades.txt File Creation operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
```



```
grades.txt
```

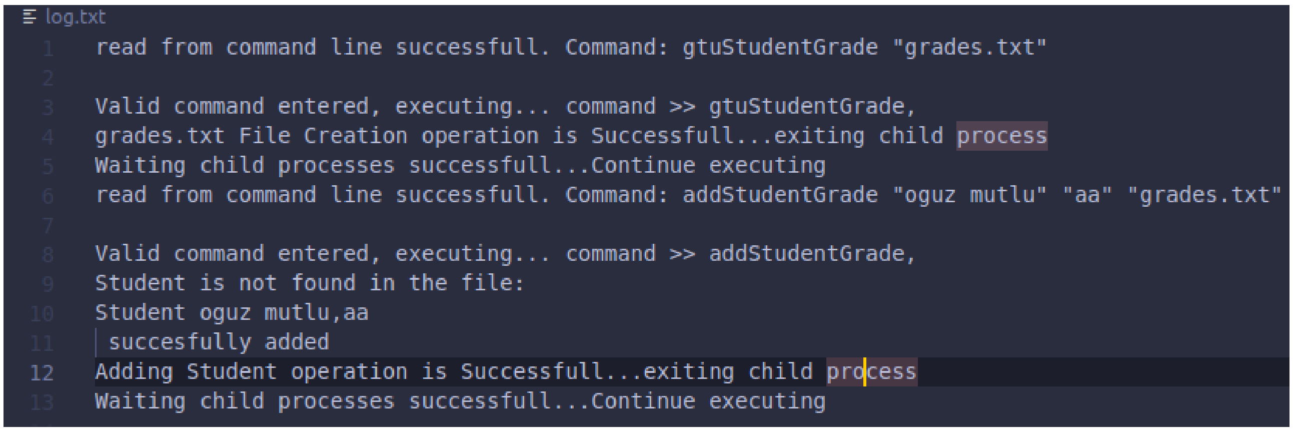
>>addStudentGrade “oguz mutlu” “aa” “grades.txt”



```
oguz@oguz-ubuntu:~/Desktop/system2024$ ./management_system
gtuStudentGrade "grades.txt"
addStudentGrade "oguz mutlu" "aa" "grades.txt"
Student is not found in the file:
```

grades.txt

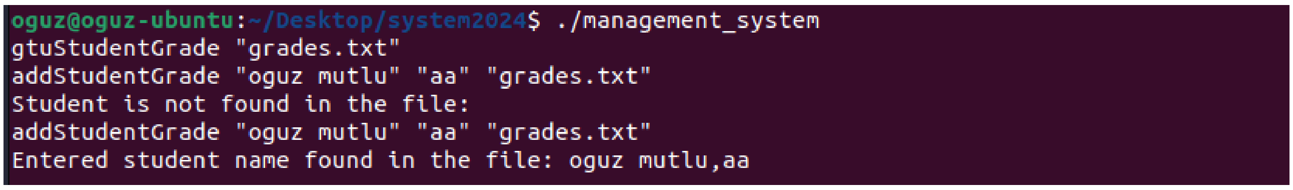
```
oguz mutlu,aa
```



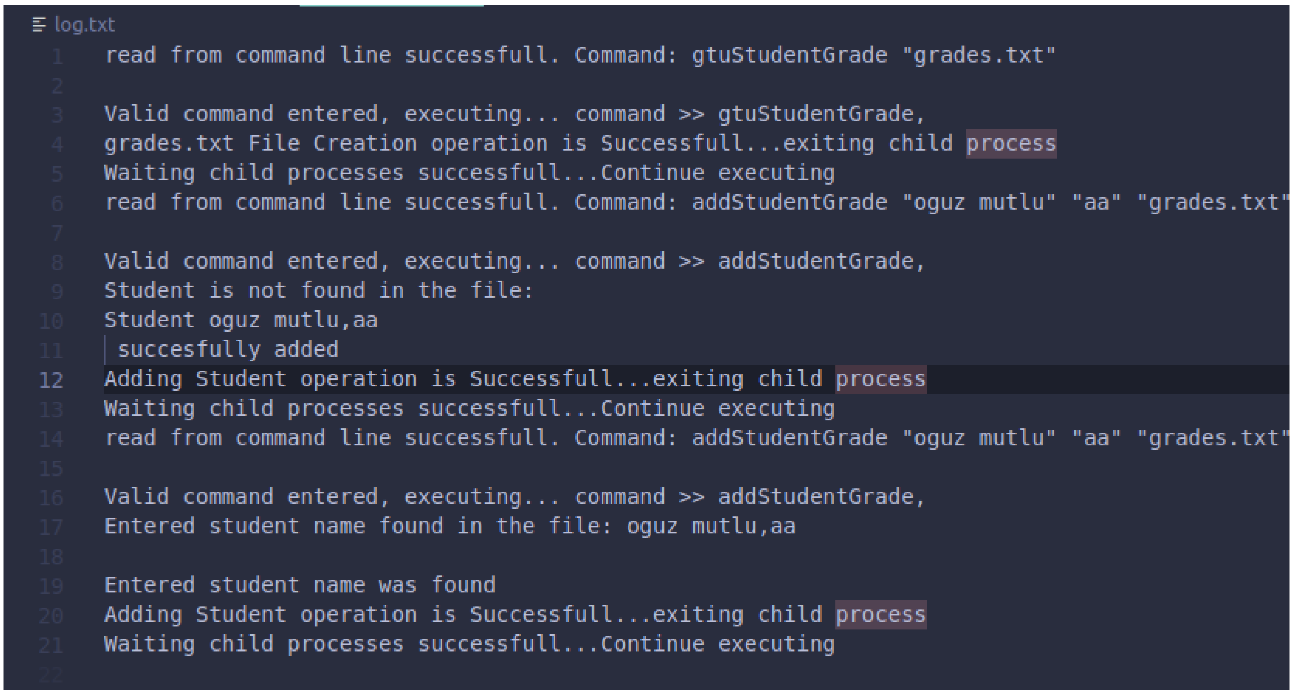
log.txt

```
read from command line successfull. Command: gtuStudentGrade "grades.txt"

Valid command entered, executing... command >> gtuStudentGrade,
grades.txt File Creation operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
read from command line successfull. Command: addStudentGrade "oguz mutlu" "aa" "grades.txt"

Valid command entered, executing... command >> addStudentGrade,
Student is not found in the file:
Student oguz mutlu,aa
 succesfully added
Adding Student operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
```

>>addStudentGrade “oguz mutlu” “aa” “grades.txt” (test existing student)

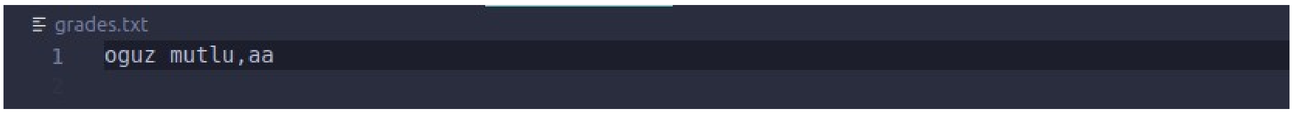


```
oguz@oguz-ubuntu:~/Desktop/system2024$ ./management_system
gtuStudentGrade "grades.txt"
addStudentGrade "oguz mutlu" "aa" "grades.txt"
Student is not found in the file:
addStudentGrade "oguz mutlu" "aa" "grades.txt"
Entered student name found in the file: oguz mutlu,aa
```



log.txt

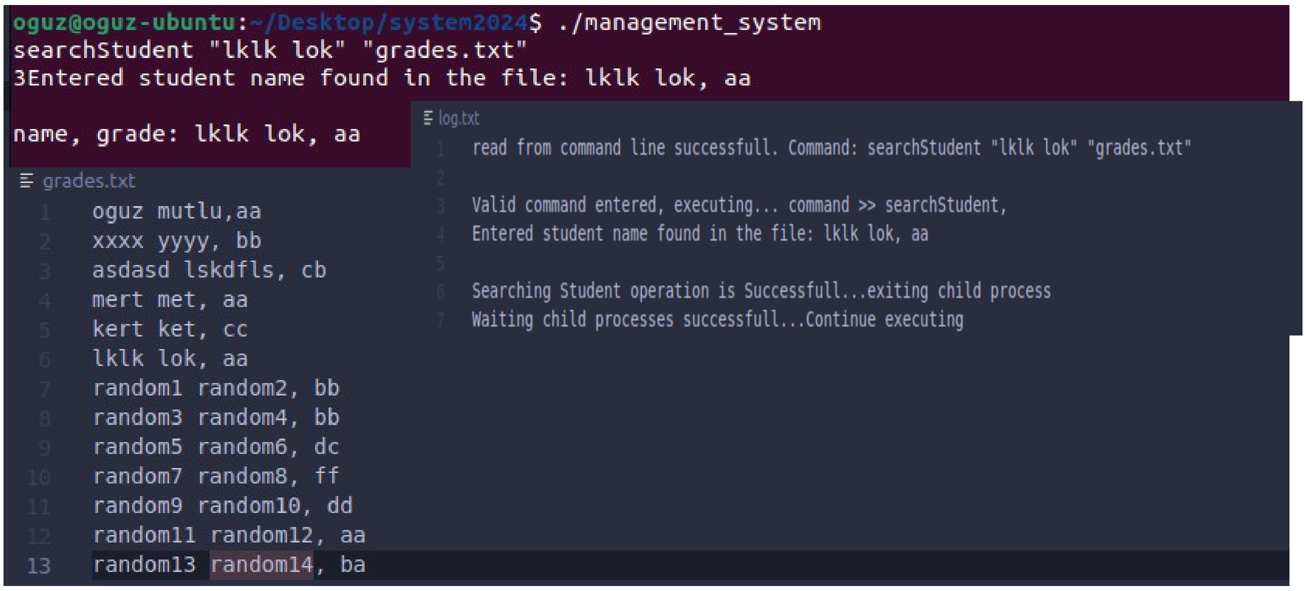
```
read from command line successfull. Command: gtuStudentGrade "grades.txt"

Valid command entered, executing... command >> gtuStudentGrade,
grades.txt File Creation operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
read from command line successfull. Command: addStudentGrade "oguz mutlu" "aa" "grades.txt"

Valid command entered, executing... command >> addStudentGrade,
Student is not found in the file:
Student oguz mutlu,aa
 succesfully added
Adding Student operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
read from command line successfull. Command: addStudentGrade "oguz mutlu" "aa" "grades.txt"

Valid command entered, executing... command >> addStudentGrade,
Entered student name found in the file: oguz mutlu,aa

Entered student name was found
Adding Student operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
```



grades.txt

```
oguz mutlu,aa
```

>>searchStudent “lklk lok” “grades.txt” (testing with existing grades.txt)



```
oguz@oguz-ubuntu:~/Desktop/system2024$ ./management_system
searchStudent "lklk lok" "grades.txt"
3Entered student name found in the file: lklk lok, aa

name, grade: lklk lok, aa
```

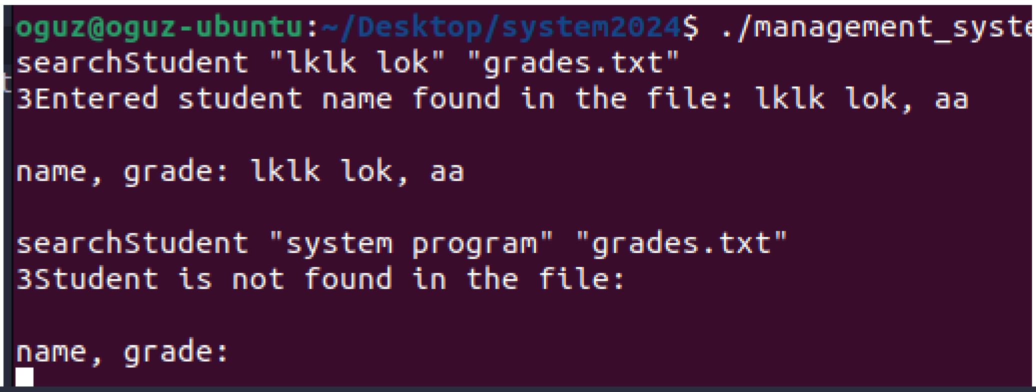
grades.txt

```
oguz mutlu,aa
xxxx yyyy, bb
asdasd lskdfls, cb
mert met, aa
kert ket, cc
lklk lok, aa
random1 random2, bb
random3 random4, bb
random5 random6, dc
random7 random8, ff
random9 random10, dd
random11 random12, aa
random13 random14, ba
```

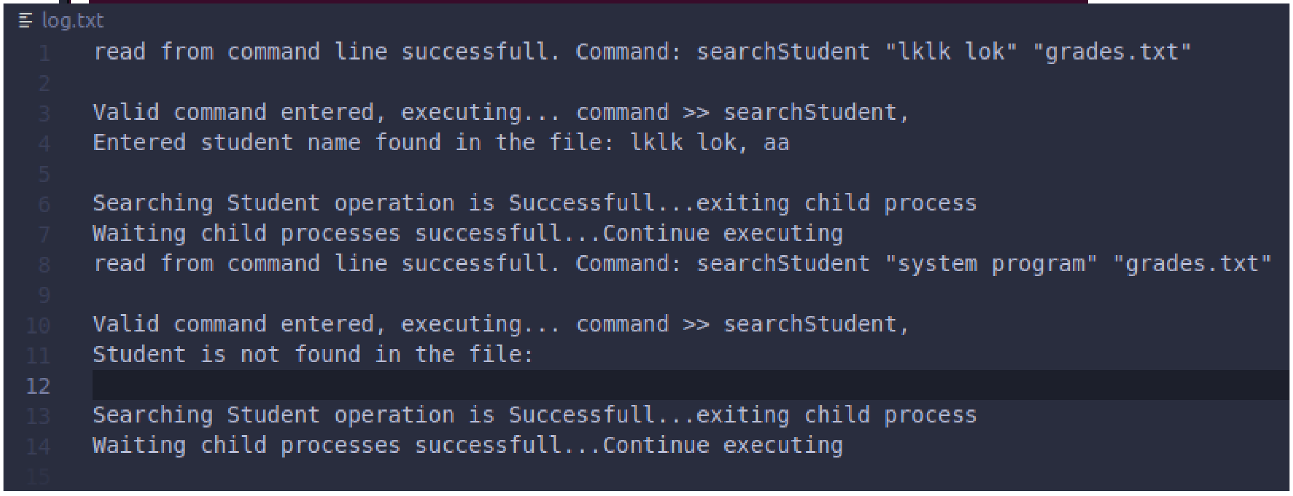
log.txt

```
read from command line successfull. Command: searchStudent "lklk lok" "grades.txt"

Valid command entered, executing... command >> searchStudent,
Entered student name found in the file: lklk lok, aa

Searching Student operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
```

>>searchStudent “system program” “grades.txt” (testing with not existing student in grades.txt)



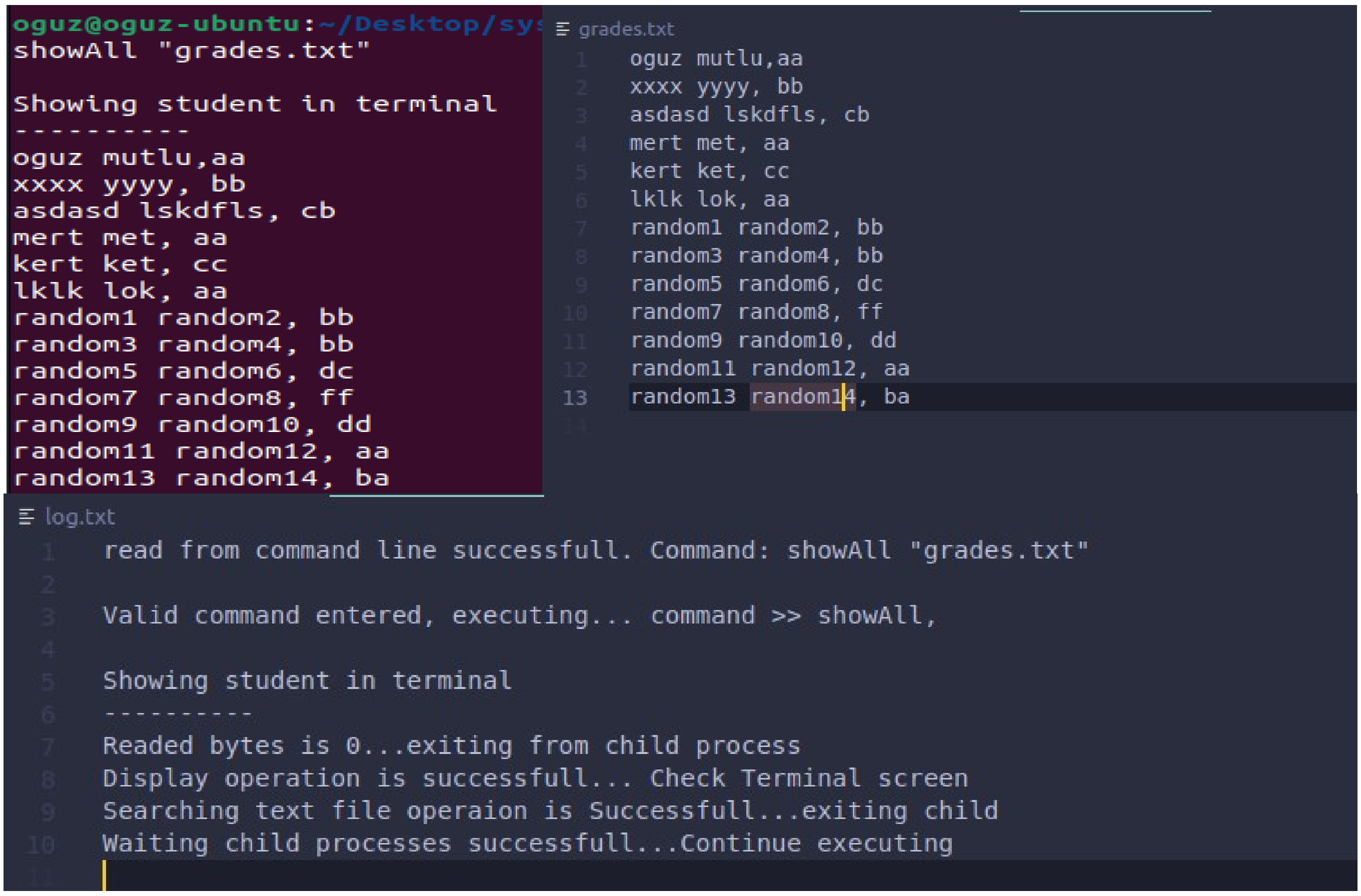
```
oguz@oguz-ubuntu:~/Desktop/system2024$ ./management_syste
searchStudent "lklk lok" "grades.txt"
3Entered student name found in the file: lklk lok, aa

name, grade: lklk lok, aa

searchStudent "system program" "grades.txt"
3Student is not found in the file:

name, grade:
```



log.txt

```
read from command line successfull. Command: searchStudent "lklk lok" "grades.txt"

Valid command entered, executing... command >> searchStudent,
Entered student name found in the file: lklk lok, aa

Searching Student operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
read from command line successfull. Command: searchStudent "system program" "grades.txt"

Valid command entered, executing... command >> searchStudent,
Student is not found in the file:

Searching Student operation is Successfull...exiting child process
Waiting child processes successfull...Continue executing
```

>>showAll “grades.txt”



```
oguz@oguz-ubuntu:~/Desktop/sys
showAll "grades.txt"

Showing student in terminal
----------
oguz mutlu,aa
xxxx yyyy, bb
asdasd lskdfls, cb
mert met, aa
kert ket, cc
lklk lok, aa
random1 random2, bb
random3 random4, bb
random5 random6, dc
random7 random8, ff
random9 random10, dd
random11 random12, aa
random13 random14, ba
```

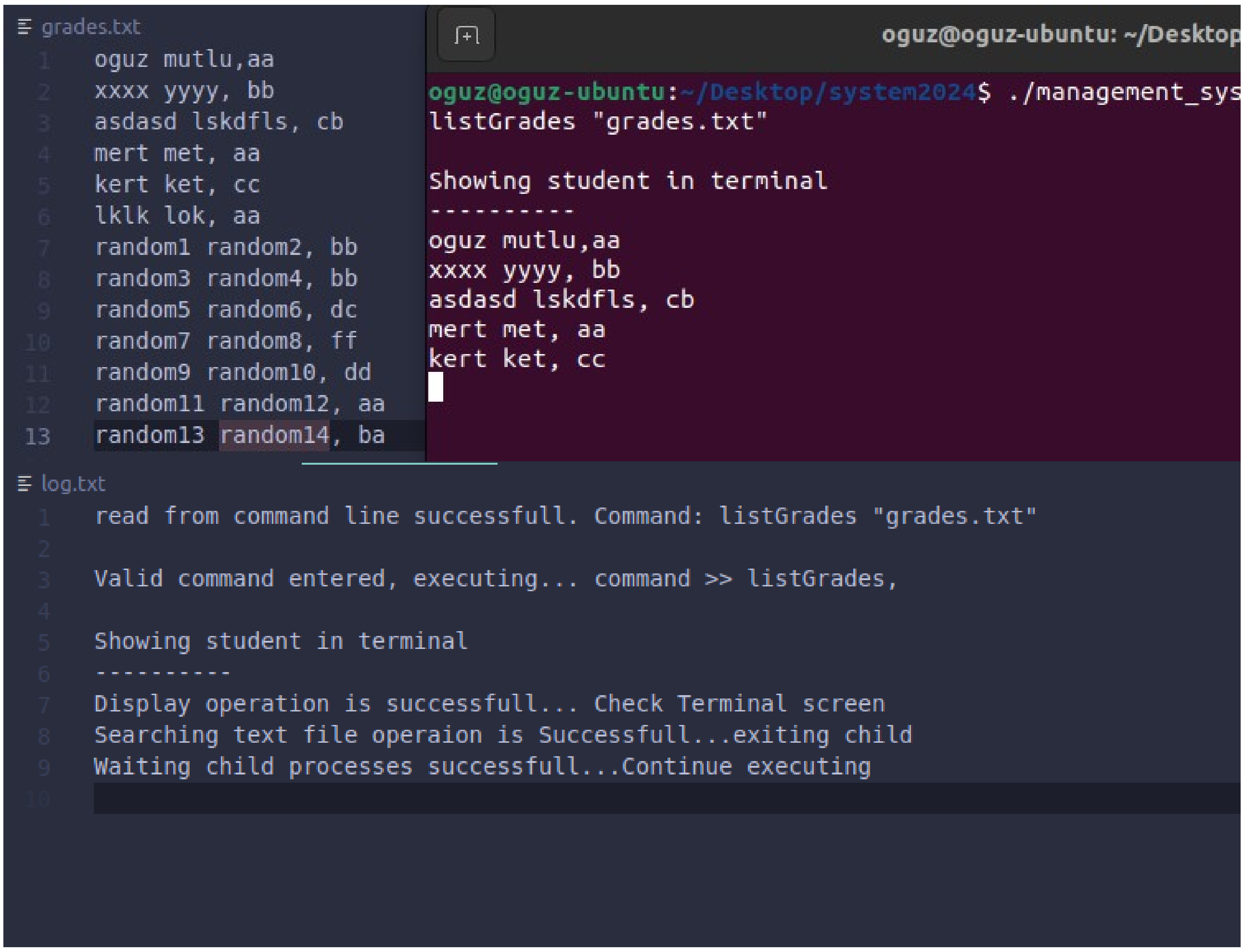
grades.txt

```
oguz mutlu,aa
xxxx yyyy, bb
asdasd lskdfls, cb
mert met, aa
kert ket, cc
lklk lok, aa
random1 random2, bb
random3 random4, bb
random5 random6, dc
random7 random8, ff
random9 random10, dd
random11 random12, aa
random13 random14, ba
```

log.txt

```
read from command line successfull. Command: showAll "grades.txt"

Valid command entered, executing... command >> showAll,

Showing student in terminal
----------
Readed bytes is 0...exiting from child process
Display operation is successfull... Check Terminal screen
Searching text file operaion is Successfull...exiting child
Waiting child processes successfull...Continue executing
```

>>listGrades “grades.txt”



grades.txt

```
oguz mutlu,aa
xxxx yyyy, bb
asdasd lskdfls, cb
mert met, aa
kert ket, cc
lklk lok, aa
random1 random2, bb
random3 random4, bb
random5 random6, dc
random7 random8, ff
random9 random10, dd
random11 random12, aa
random13 random14, ba
```

oguz@oguz-ubuntu: ~/Desktop

```
oguz@oguz-ubuntu:~/Desktop/system2024$ ./management_sys
listGrades "grades.txt"

Showing student in terminal
----------
oguz mutlu,aa
xxxx yyyy, bb
asdasd lskdfls, cb
mert met, aa
kert ket, cc
```

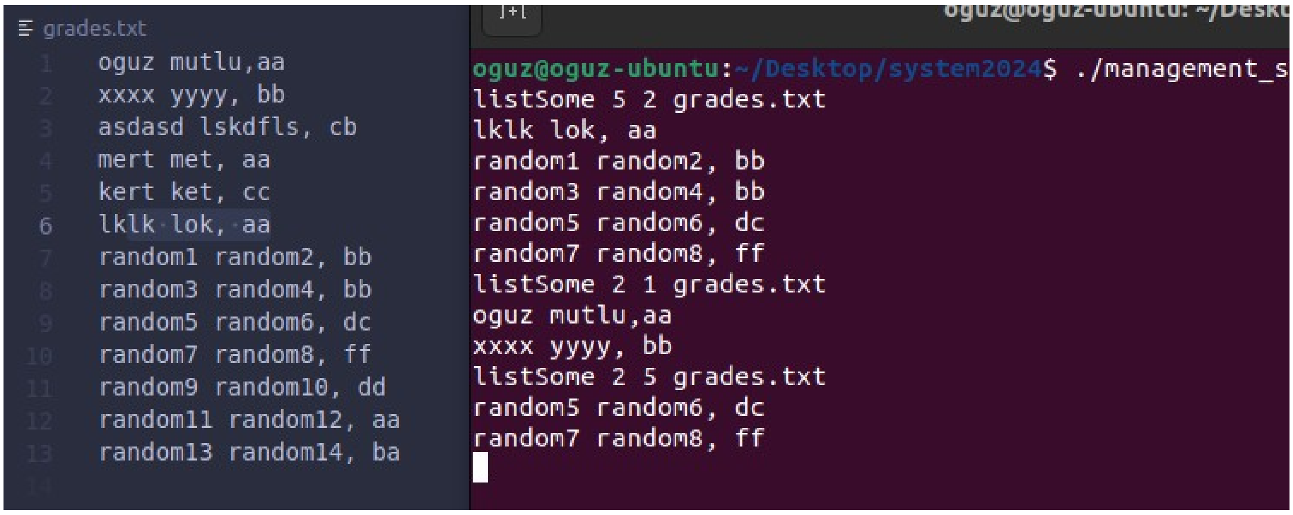
log.txt

```
read from command line successfull. Command: listGrades "grades.txt"

Valid command entered, executing... command >> listGrades,

Showing student in terminal
----------
Display operation is successfull... Check Terminal screen
Searching text file operaion is Successfull...exiting child
Waiting child processes successfull...Continue executing
```

>>listSome 5 2 “grades.txt”

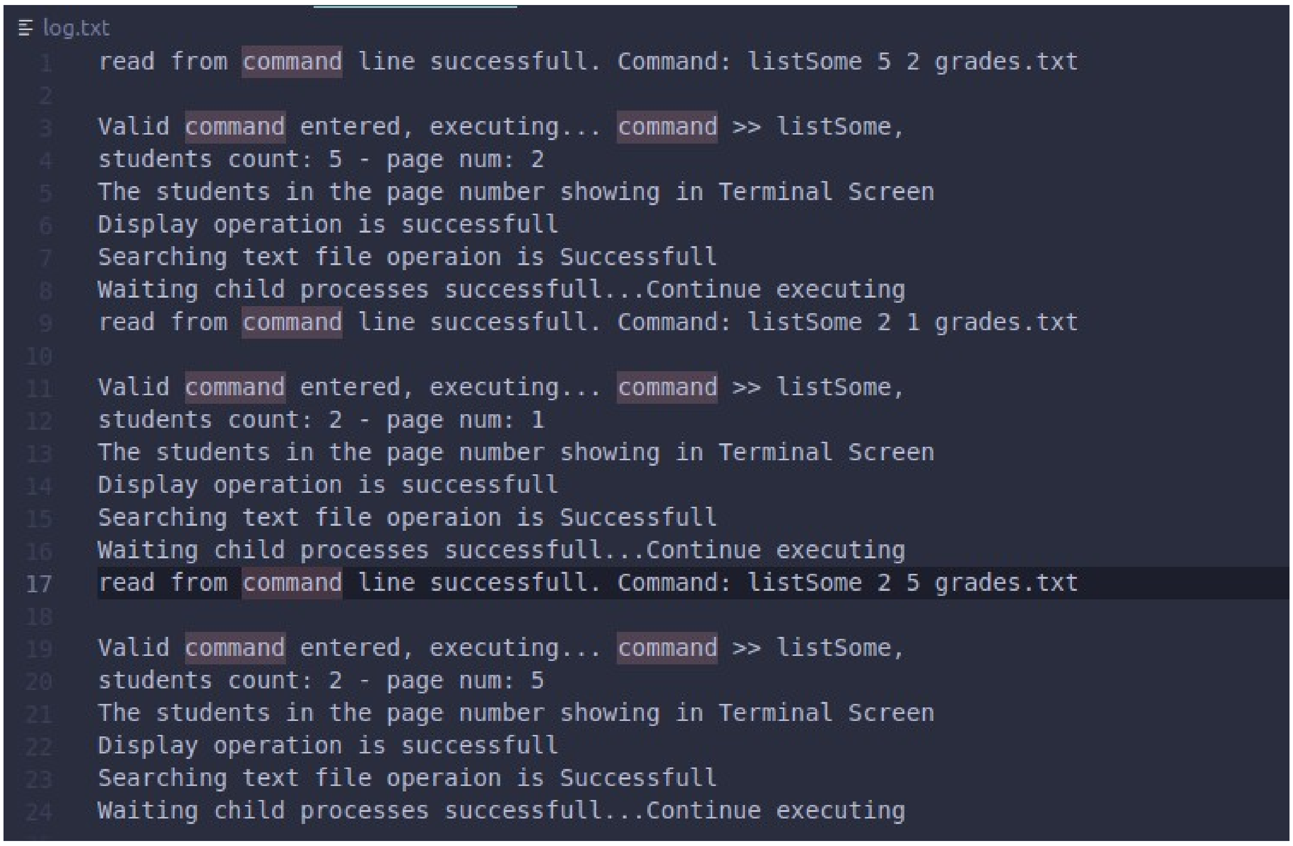
>>listSome 2 1 “grades.txt”

>>listSome 2 5 grades.txt



```
oguz mutlu,aa
xxxx yyyy, bb
asdasd lskdfls, cb
mert met, aa
kert ket, cc
lklk lok, aa
random1 random2, bb
random3 random4, bb
random5 random6, dc
random7 random8, ff
random9 random10, dd
random11 random12, aa
random13 random14, ba
```

```
oguz@oguz-ubuntu:~/Desktop/system2024$ ./management_s
listSome 5 2 grades.txt
lklk lok, aa
random1 random2, bb
random3 random4, bb
random5 random6, dc
random7 random8, ff
listSome 2 1 grades.txt
oguz mutlu,aa
xxxx yyyy, bb
listSome 2 5 grades.txt
random5 random6, dc
random7 random8, ff
```



```
read from command line successfull. Command: listSome 5 2 grades.txt

Valid command entered, executing... command >> listSome,
students count: 5 - page num: 2
The students in the page number showing in Terminal Screen
Display operation is successfull
Searching text file operaion is Successfull
Waiting child processes successfull...Continue executing
read from command line successfull. Command: listSome 2 1 grades.txt

Valid command entered, executing... command >> listSome,
students count: 2 - page num: 1
The students in the page number showing in Terminal Screen
Display operation is successfull
Searching text file operaion is Successfull
Waiting child processes successfull...Continue executing
read from command line successfull. Command: listSome 2 5 grades.txt

Valid command entered, executing... command >> listSome,
students count: 2 - page num: 5
The students in the page number showing in Terminal Screen
Display operation is successfull
Searching text file operaion is Successfull
Waiting child processes successfull...Continue executing
```

>>sortAll “grades.txt”

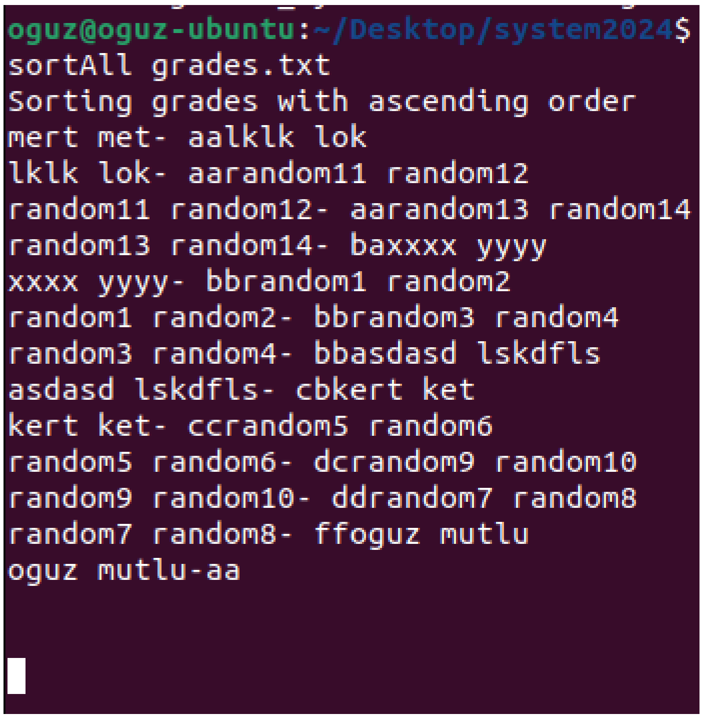
>>sortAll n grades.txt

>>sortAll g grades.txt

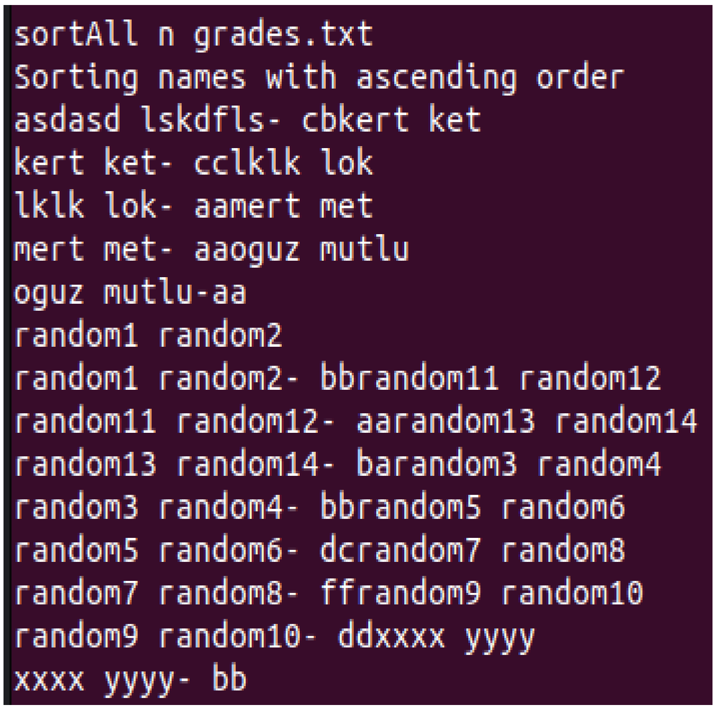
>>sortAll d grades.txt

>>sortAll c grades.txt

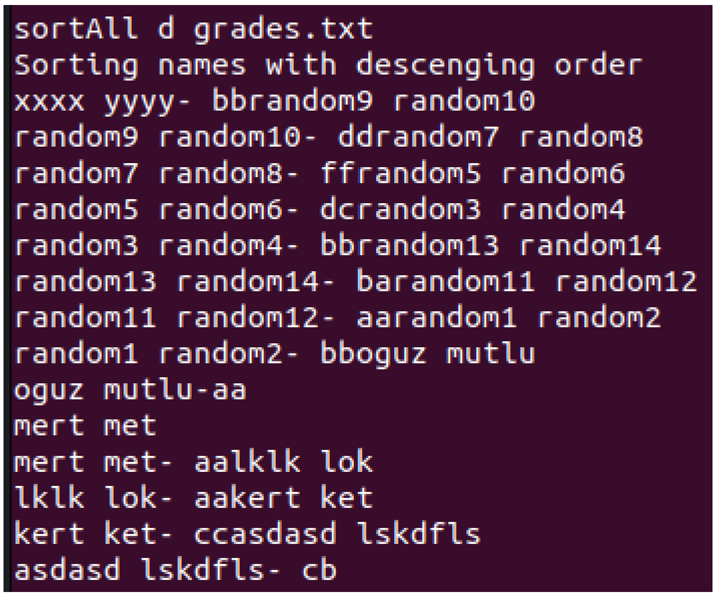
NOTE TO TA : I think the sort function actually works properly. Even though I did the null character termination, the function could not detect the null character and reset the array, so it prints the two arrays combined. It does not give proper output, but if you examine it carefully up to the grade part, it performs the sorting operation.The top side sorting is shown as ascending, the bottom side sorting is shown as descending.



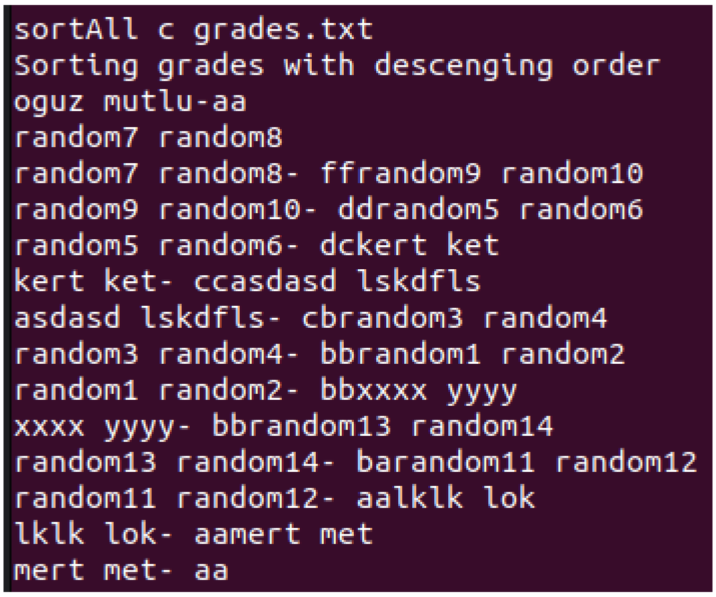
```
oguz@oguz-ubuntu:~/Desktop/system2024$
sortAll grades.txt
Sorting grades with ascending order
mert met- aalklk lok
lklk lok- aarandom11 random12
random11 random12- aarandom13 random14
random13 random14- baxxxx yyyy
xxxx yyyy- bbrandom1 random2
random1 random2- bbrandom3 random4
random3 random4- bbasdasd lskdfls
asdasd lskdfls- cbkert ket
kert ket- ccrandom5 random6
random5 random6- dcrandom9 random10
random9 random10- ddrandom7 random8
random7 random8- ffoguz mutlu
oguz mutlu-aa
```



```
sortAll n grades.txt
Sorting names with ascending order
asdasd lskdfls- cbkert ket
kert ket- cclklk lok
lklk lok- aamert met
mert met- aaoguz mutlu
oguz mutlu-aa
random1 random2
random1 random2- bbrandom11 random12
random11 random12- aarandom13 random14
random13 random14- barandom3 random4
random3 random4- bbrandom5 random6
random5 random6- dcrandom7 random8
random7 random8- ffrandom9 random10
random9 random10- ddxxxx yyyy
xxxx yyyy- bb
```



```
sortAll d grades.txt
Sorting names with descenging order
xxxx yyyy- bbrandom9 random10
random9 random10- ddrandom7 random8
random7 random8- ffrandom5 random6
random5 random6- dcrandom3 random4
random3 random4- bbrandom13 random14
random13 random14- barandom11 random12
random11 random12- aarandom1 random2
random1 random2- bboguz mutlu
oguz mutlu-aa
mert met
mert met- aalklk lok
lklk lok- aakert ket
kert ket- ccasdasd lskdfls
asdasd lskdfls- cb
```



```
sortAll c grades.txt
Sorting grades with descenging order
oguz mutlu-aa
random7 random8
random7 random8- ffrandom9 random10
random9 random10- ddrandom5 random6
random5 random6- dckert ket
kert ket- ccasdasd lskdfls
asdasd lskdfls- cbrandom3 random4
random3 random4- bbrandom1 random2
random1 random2- bbxxxx yyyy
xxxx yyyy- bbrandom13 random14
random13 random14- barandom11 random12
random11 random12- aalklk lok
lklk lok- aamert met
mert met- aa
```