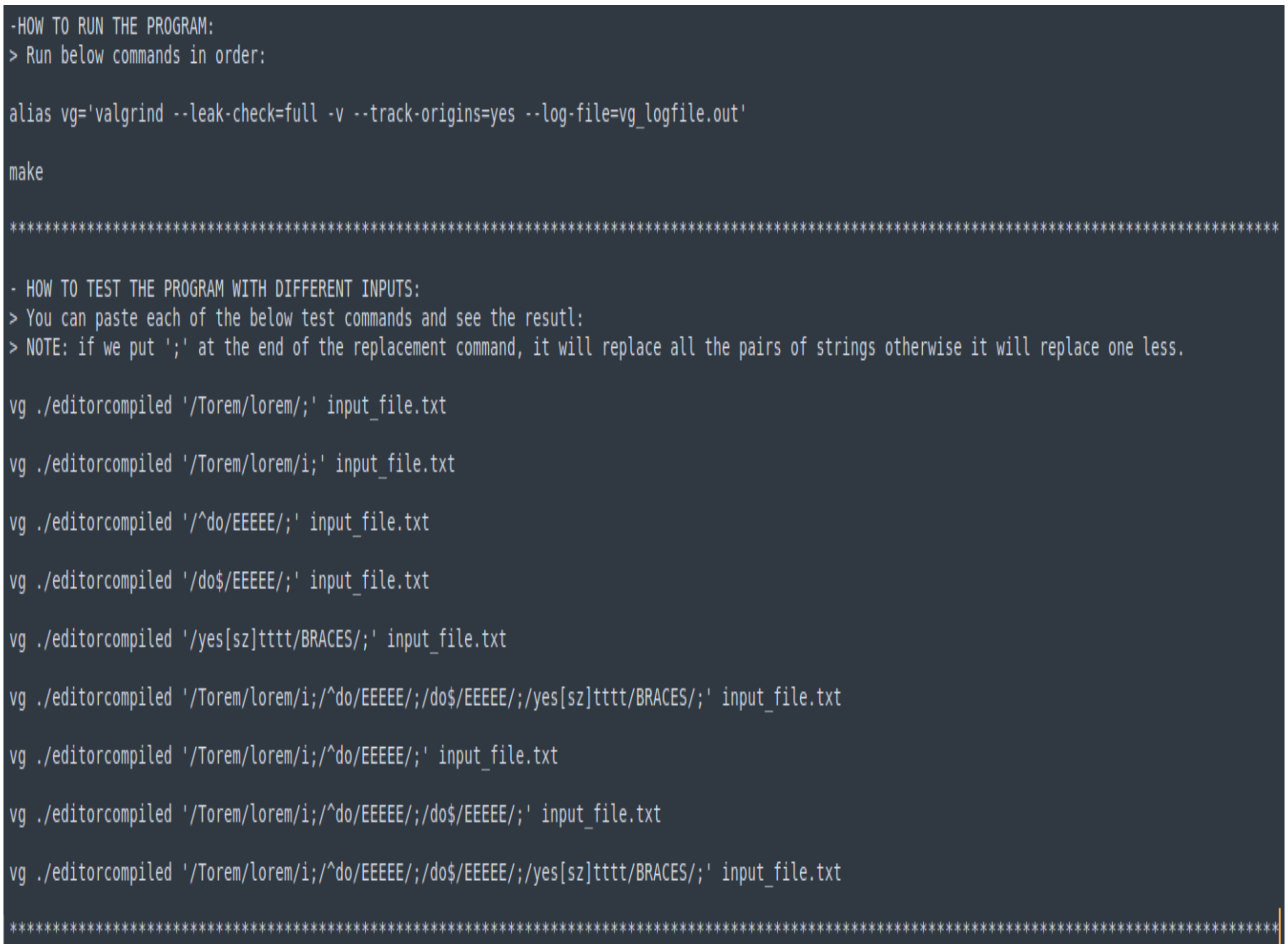
Systems Programming HW1 Report
Mohammad Ashraf Yawar 161044123
March 18, 2022

- HOW TO RUN AND TEST THE PROGRAM WITH DIFFERENT INPUTS?
- Working DEMO link: https://youtu.be/tuDPm51PFtI

- You can find the instructions in README.txt in order to run and test the program.



```
-HOW TO RUN THE PROGRAM:
> Run below commands in order:

alias vg='valgrind --leak-check=full -v --track-origins=yes --log-file=vg_logfile.out'

make

******************************************************************************************************************

- HOW TO TEST THE PROGRAM WITH DIFFERENT INPUTS:
> You can paste each of the below test commands and see the resutl:
> NOTE: if we put ';' at the end of the replacement command, it will replace all the pairs of strings otherwise it will replace one less.

vg ./editorcompiled '/Torem/lorem/;' input_file.txt

vg ./editorcompiled '/Torem/lorem/i;' input_file.txt

vg ./editorcompiled '/^do/EEEEE/;' input_file.txt

vg ./editorcompiled '/do$/EEEEE/;' input_file.txt

vg ./editorcompiled '/yes[sz]tttt/BRACES/;' input_file.txt

vg ./editorcompiled '/Torem/lorem/i;/^do/EEEEE/;/do$/EEEEE/;/yes[sz]tttt/BRACES/;' input_file.txt

vg ./editorcompiled '/Torem/lorem/i;/^do/EEEEE/;' input_file.txt

vg ./editorcompiled '/Torem/lorem/i;/^do/EEEEE/;/do$/EEEEE/;' input_file.txt

vg ./editorcompiled '/Torem/lorem/i;/^do/EEEEE/;/do$/EEEEE/;/yes[sz]tttt/BRACES/;' input_file.txt

******************************************************************************************************************
```

Implemented Concepts:

- File read, write, lock using syscalls.
- Temporary files using mkstemp() syscall.
- Regular expression.
- Make files.

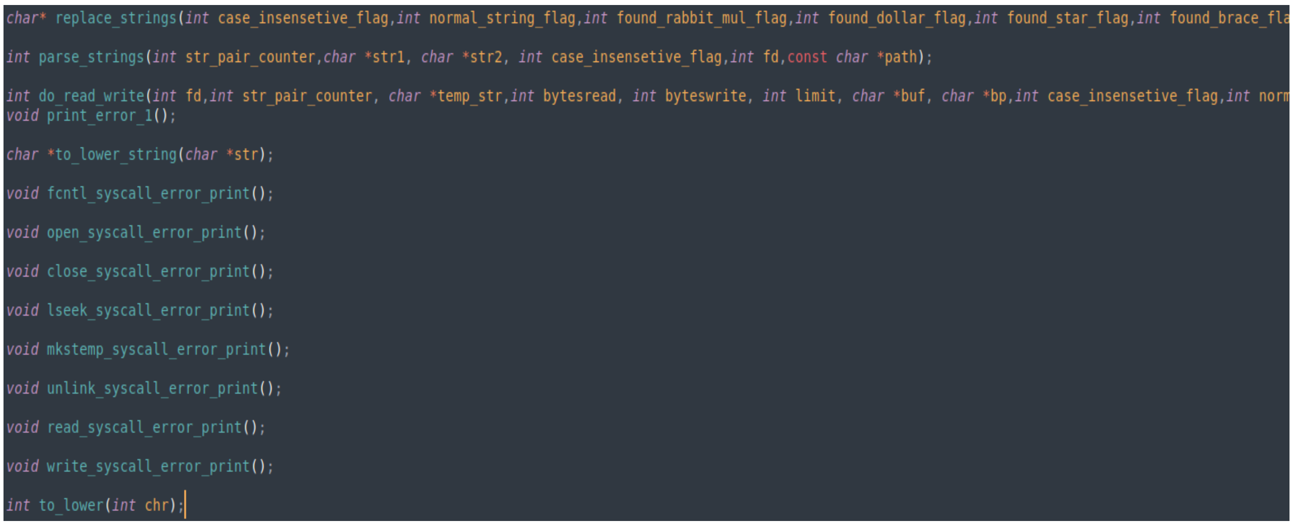
Working Cases:
- This program works for cases **a,b,c,d,e** and **f.**
- Works on **relative path**.

Note Working Cases:

- This program does not work for cases **g** and the all **combined** case.
- Does not work on **absolute path**.

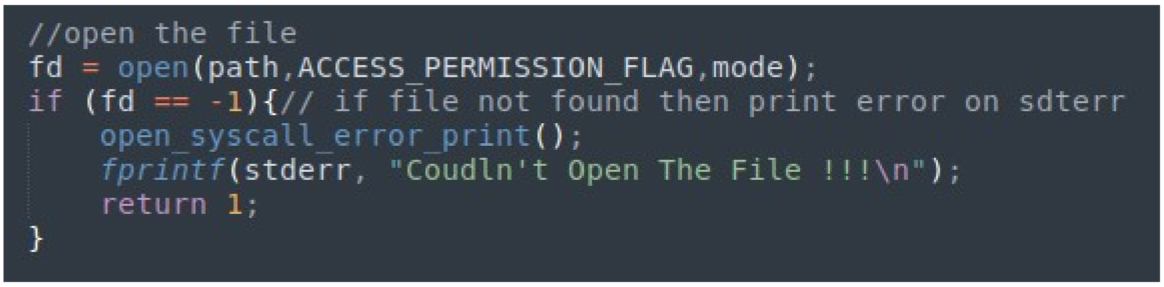
Design Explanation:

- I separated my program into functions using make file where each function does a specific task:



```c
char* replace_strings(int case_insensetive_flag,int normal_string_flag,int found_rabbit_mul_flag,int found_dollar_flag,int found_star_flag,int found_brace_fla

int parse_strings(int str_pair_counter,char *str1, char *str2, int case_insensetive_flag,int fd,const char *path);

int do_read_write(int fd,int str_pair_counter, char *temp_str,int bytesread, int byteswrite, int limit, char *buf, char *bp,int case_insensetive_flag,int norm
void print_error_1();

char *to_lower_string(char *str);

void fcntl_syscall_error_print();

void open_syscall_error_print();

void close_syscall_error_print();

void lseek_syscall_error_print();

void mkstemp_syscall_error_print();

void unlink_syscall_error_print();

void read_syscall_error_print();

void write_syscall_error_print();

int to_lower(int chr);
```

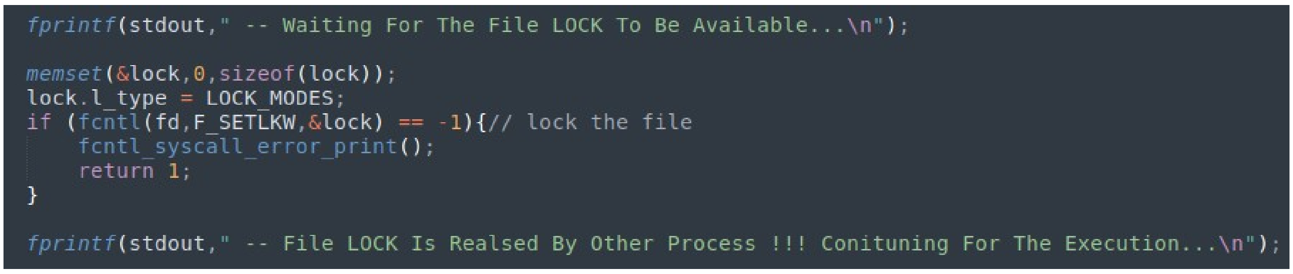
- On main function I start the the program procedure by opening the file and checking all the necessary error and act accordingly.



```c
//open the file
fd = open(path,ACCESS_PERMISSION_FLAG,mode);
if (fd == -1){// if file not found then print error on sdterr
    open_syscall_error_print();
    fprintf(stderr, "Coudln't Open The File !!!\n");
    return 1;
}
```

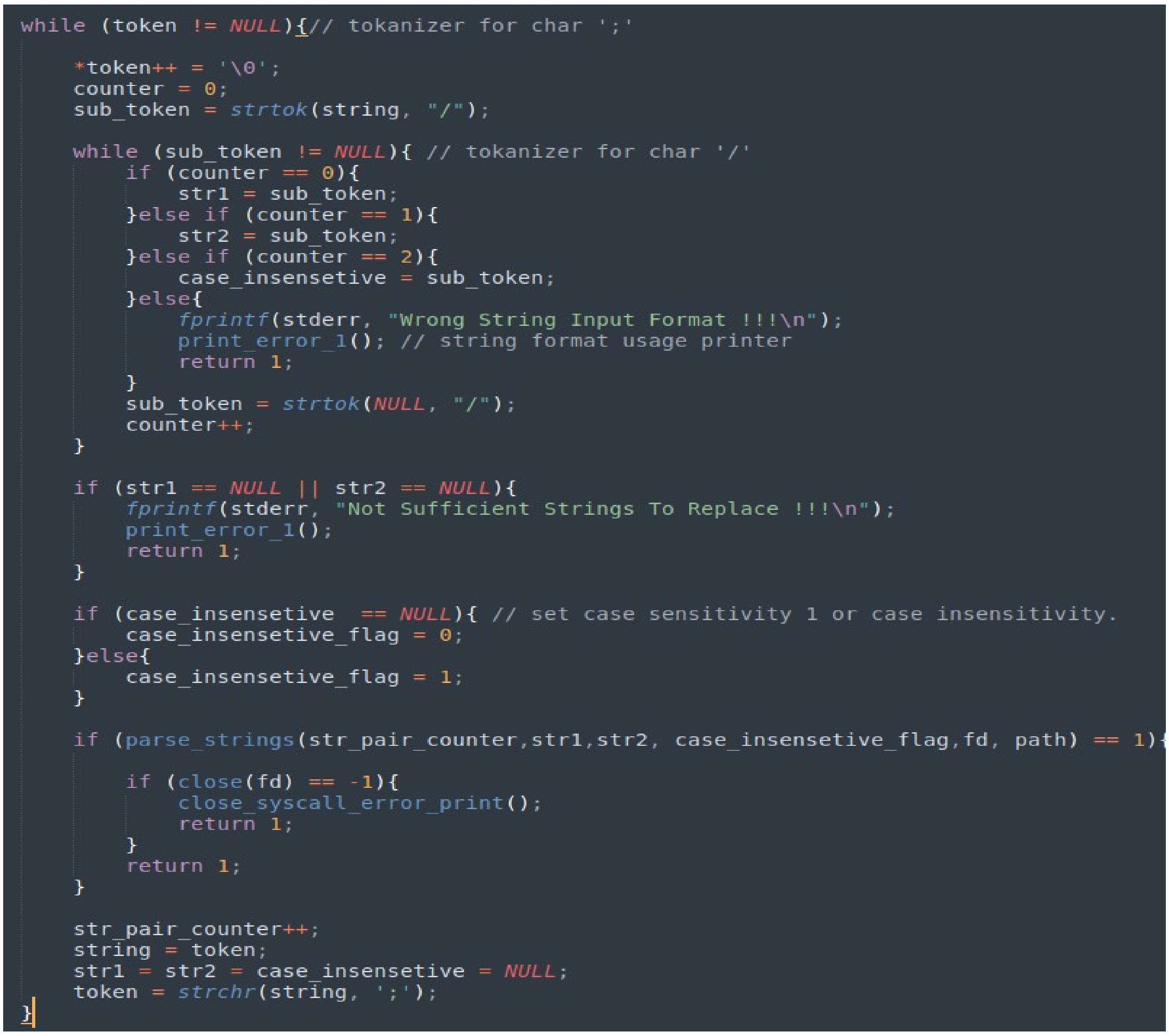
- All the System-Calls and their possible return error values are checked with detailed errno checks.

- After opening the file, the file is locked using fcntl() syscall, if the file is already locked, the process waits until the file become available, I added some informative strings using fprintf into stdout so that when ever we run the program on terminal , we can observe and see what is happening in which stage.



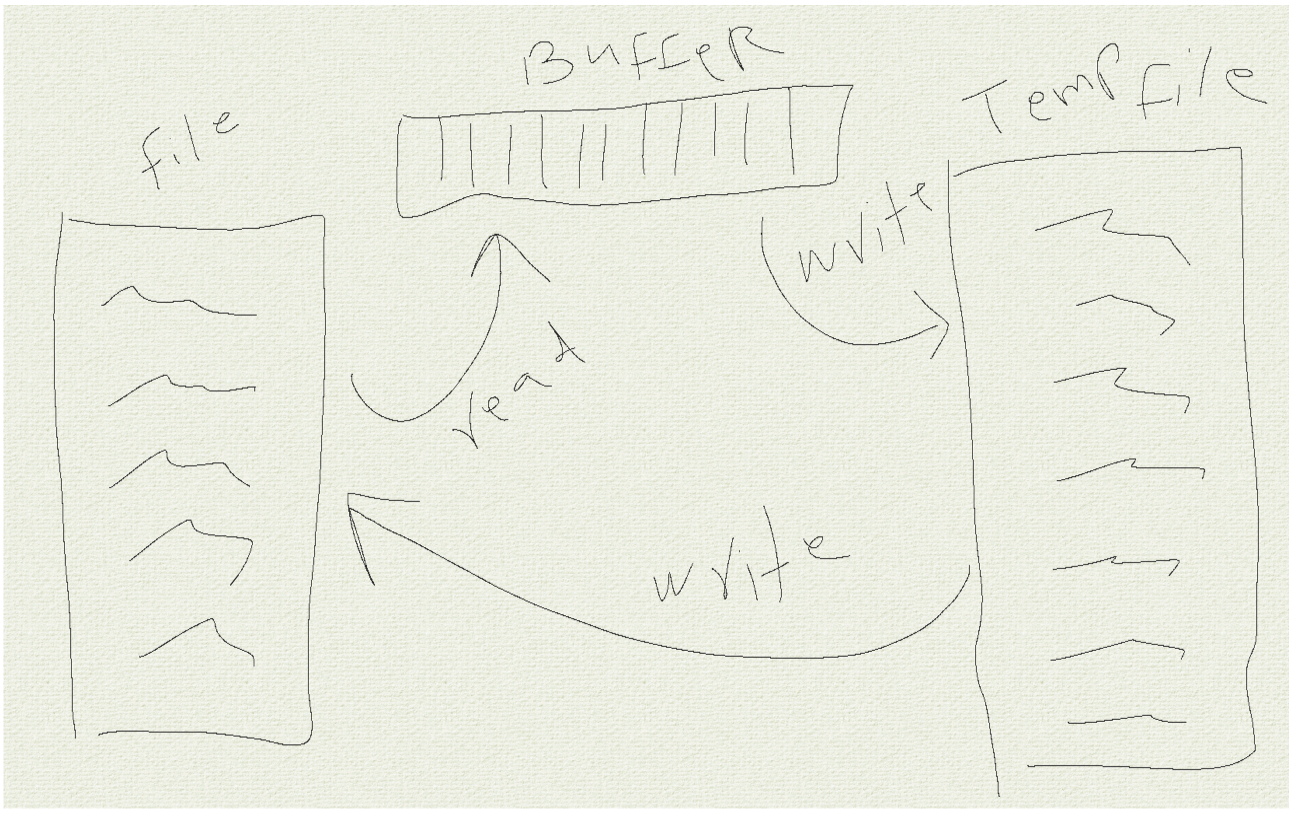
```c
fprintf(stdout," -- Waiting For The File LOCK To Be Available...\n");

memset(&lock,0,sizeof(lock));
lock.l_type = LOCK_MODES;
if (fcntl(fd,F_SETLKW,&lock) == -1){// lock the file
    fcntl_syscall_error_print();
    return 1;
}

fprintf(stdout," -- File LOCK Is Realsed By Other Process !!! Conituning For The Execution...\n");
```

- After the file is locked I parse the string to be replaced and the string to replace according the working cases.



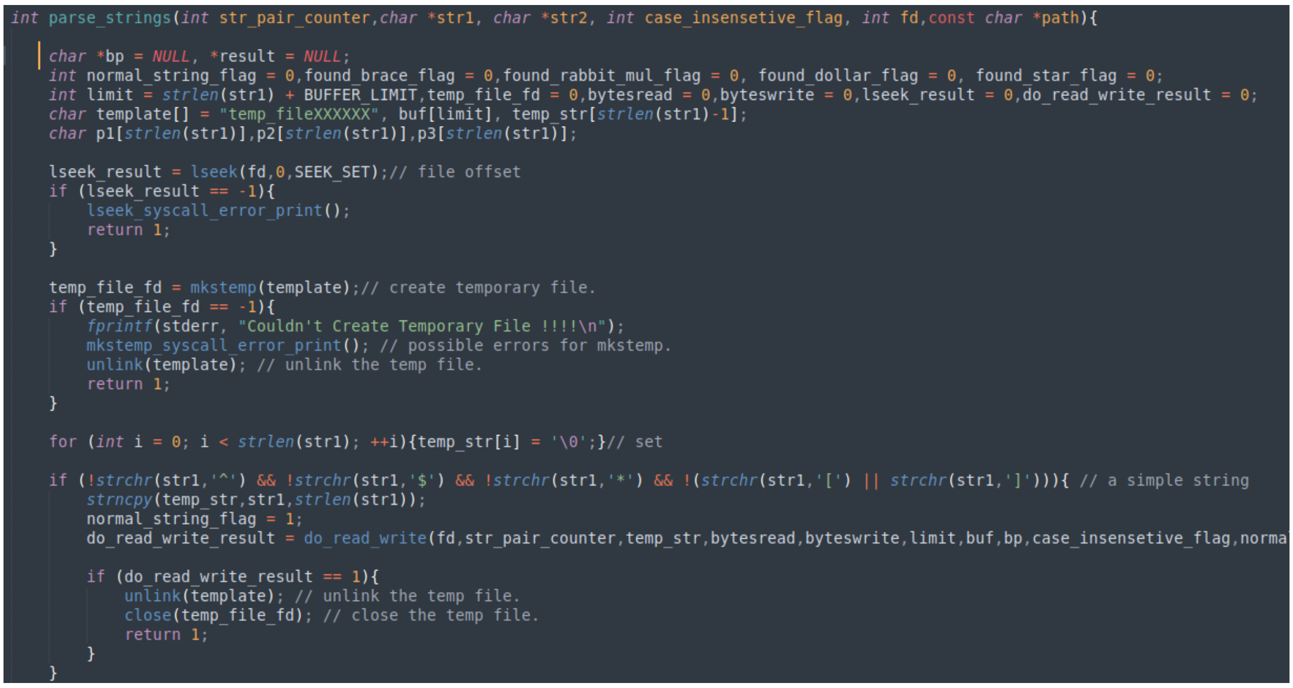
```c
while (token != NULL){// tokanizer for char ';'

    *token++ = '\0';
    counter = 0;
    sub_token = strtok(string, "/");

    while (sub_token != NULL){ // tokanizer for char '/'
        if (counter == 0){
            str1 = sub_token;
        }else if (counter == 1){
            str2 = sub_token;
        }else if (counter == 2){
            case_insensetive = sub_token;
        }else{
            fprintf(stderr, "Wrong String Input Format !!!\n");
            print_error_1(); // string format usage printer
            return 1;
        }
        sub_token = strtok(NULL, "/");
        counter++;
    }

    if (str1 == NULL || str2 == NULL){
        fprintf(stderr, "Not Sufficient Strings To Replace !!!\n");
        print_error_1();
        return 1;
    }

    if (case_insensetive  == NULL){ // set case sensitivity 1 or case insensitivity.
        case_insensetive_flag = 0;
    }else{
        case_insensetive_flag = 1;
    }

    if (parse_strings(str_pair_counter,str1,str2, case_insensetive_flag,fd, path) == 1){

        if (close(fd) == -1){
            close_syscall_error_print();
            return 1;
        }
        return 1;
    }

    str_pair_counter++;
    string = token;
    str1 = str2 = case_insensetive = NULL;
    token = strchr(string, ';');
}
```

- At a glance I set a buffer size in this case 5000 bytes to read it as bulk from the file, I read the 5000 bytes into file in each read call until the end of file is reached, after that I have 5000 char to process and do replacement If necessary, and I create a temporary file using mkstemp() syscall and store the manipulated and process buffer into temp file and continues like this until all the chars inside original file is read and process and stored in temp file, at the end I write all the temp file into original file and override it with new processed and replaced data.



- Inside parse_string() function:

- Set the file offset to the beginning of the file.
- Open temp file.
- Parse the first string for regex and set some necessary flag to process it later on.
- Call do_read_write() function.
- Close and free necessary variables and pointers.

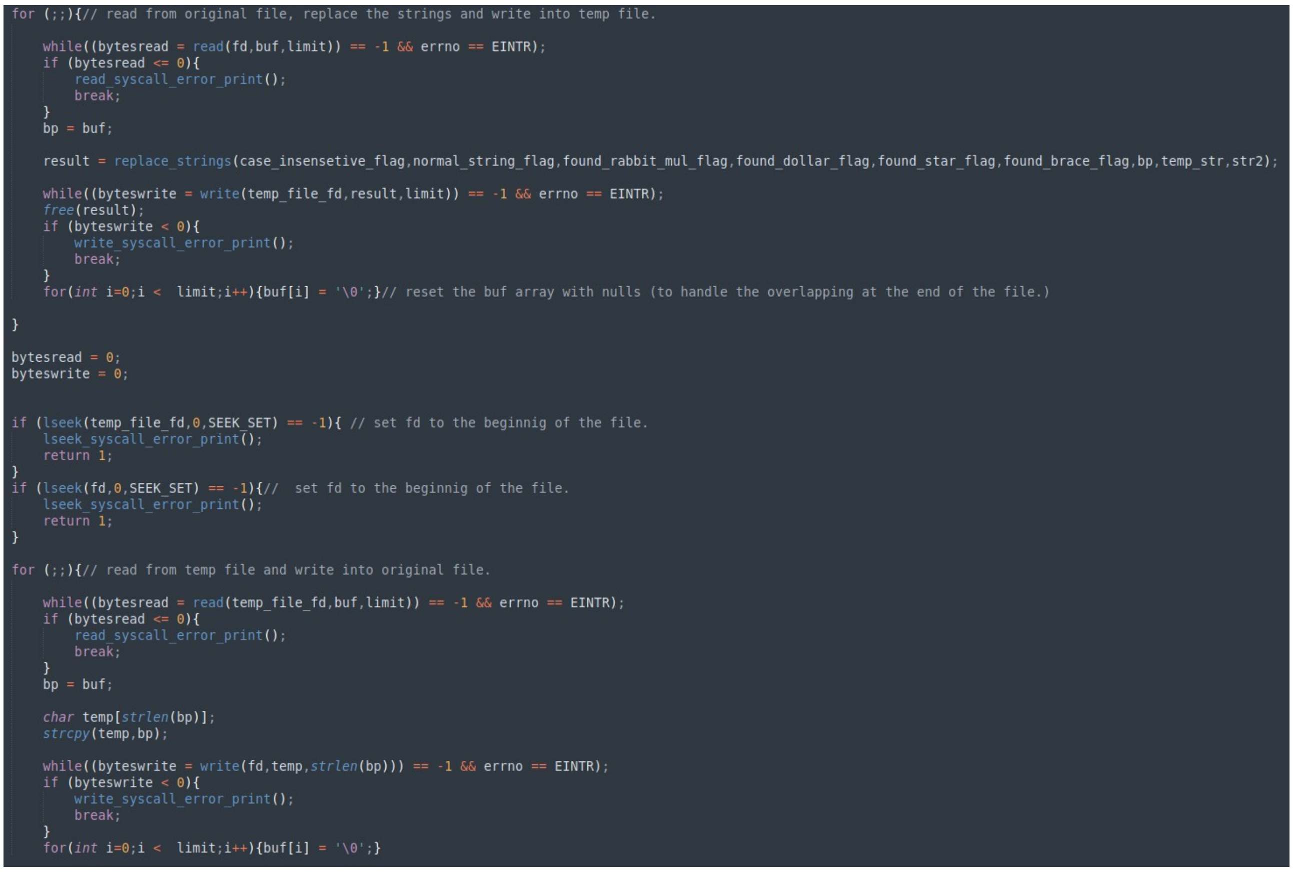


```c
int parse_strings(int str_pair_counter,char *str1, char *str2, int case_insensetive_flag, int fd,const char *path){

    char *bp = NULL, *result = NULL;
    int normal_string_flag = 0,found_brace_flag = 0,found_rabbit_mul_flag = 0, found_dollar_flag = 0, found_star_flag = 0;
    int limit = strlen(str1) + BUFFER_LIMIT,temp_file_fd = 0,bytesread = 0,byteswrite = 0,lseek_result = 0,do_read_write_result = 0;
    char template[] = "temp_fileXXXXXX", buf[limit], temp_str[strlen(str1)-1];
    char p1[strlen(str1)],p2[strlen(str1)],p3[strlen(str1)];

    lseek_result = lseek(fd,0,SEEK_SET);// file offset
    if (lseek_result == -1){
        lseek_syscall_error_print();
        return 1;
    }

    temp_file_fd = mkstemp(template);// create temporary file.
    if (temp_file_fd == -1){
        fprintf(stderr, "Couldn't Create Temporary File !!!!\n");
        mkstemp_syscall_error_print(); // possible errors for mkstemp.
        unlink(template); // unlink the temp file.
        return 1;
    }

    for (int i = 0; i < strlen(str1); ++i){temp_str[i] = '\0';}// set

    if (!strchr(str1,'^') && !strchr(str1,'$') && !strchr(str1,'*') && !(strchr(str1,'[') || strchr(str1,']'))){ // a simple string
        strncpy(temp_str,str1,strlen(str1));
        normal_string_flag = 1;
        do_read_write_result = do_read_write(fd,str_pair_counter,temp_str,bytesread,byteswrite,limit,buf,bp,case_insensetive_flag,norma

        if (do_read_write_result == 1){
            unlink(template); // unlink the temp file.
            close(temp_file_fd); // close the temp file.
            return 1;
        }
    }
```

- **NOTE**: I put some informative prints on console so that we can keep track and observe the program executions and it's steps at a glance.

- Inside do_read_write() function:

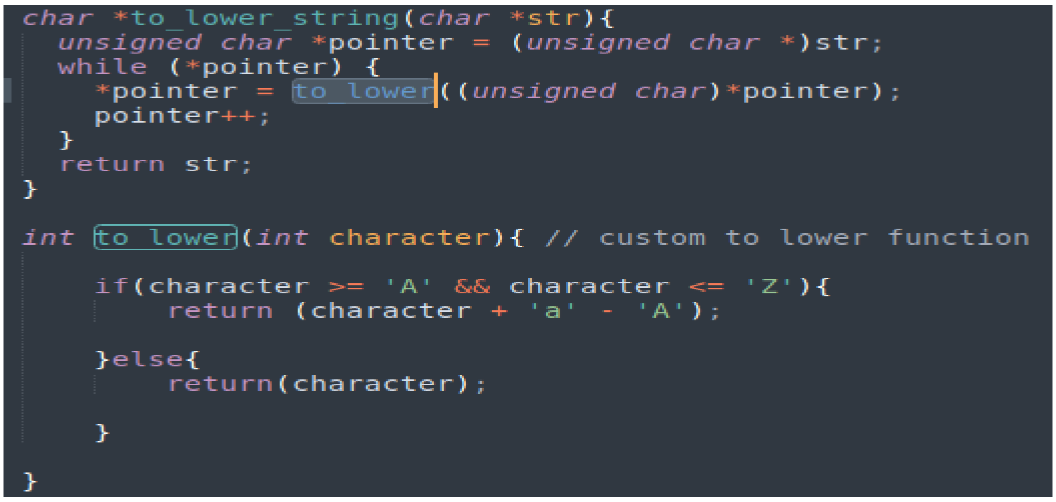
- In a continues for loop:
- Read data from original file into buffer.
- Processs it and replace the neccesary strings and create a newly replaces string.
- Write the proccessed string into temporary file.
- When all the chars are processed in original file and put into temporary file THEN: copy temporary file content into original file.
- Close the temporary file.



```c
for (;;){// read from original file, replace the strings and write into temp file.

    while((bytesread = read(fd,buf,limit)) == -1 && errno == EINTR);
    if (bytesread <= 0){
        read_syscall_error_print();
        break;
    }
    bp = buf;

    result = replace_strings(case_insensetive_flag,normal_string_flag,found_rabbit_mul_flag,found_dollar_flag,found_star_flag,found_brace_flag,bp,temp_str,str2);

    while((byteswrite = write(temp_file_fd,result,limit)) == -1 && errno == EINTR);
    free(result);
    if (byteswrite < 0){
        write_syscall_error_print();
        break;
    }
    for(int i=0;i <  limit;i++){buf[i] = '\0';}// reset the buf array with nulls (to handle the overlapping at the end of the file.)

}

bytesread = 0;
byteswrite = 0;


if (lseek(temp_file_fd,0,SEEK_SET) == -1){ // set fd to the beginnig of the file.
    lseek_syscall_error_print();
    return 1;
}
if (lseek(fd,0,SEEK_SET) == -1){//  set fd to the beginnig of the file.
    lseek_syscall_error_print();
    return 1;
}

for (;;){// read from temp file and write into original file.

    while((bytesread = read(temp_file_fd,buf,limit)) == -1 && errno == EINTR);
    if (bytesread <= 0){
        read_syscall_error_print();
        break;
    }
    bp = buf;

    char temp[strlen(bp)];
    strcpy(temp,bp);

    while((byteswrite = write(fd,temp,strlen(bp))) == -1 && errno == EINTR);
    if (byteswrite < 0){
        write_syscall_error_print();
        break;
    }
    for(int i=0;i <  limit;i++){buf[i] = '\0';}
```

- Inside replace_string() function:

- This function replaces the given str1 with str2 if it’s inside the give buffer string.
- I have checked all the possible errors in all the syscalls that I have used in this program.



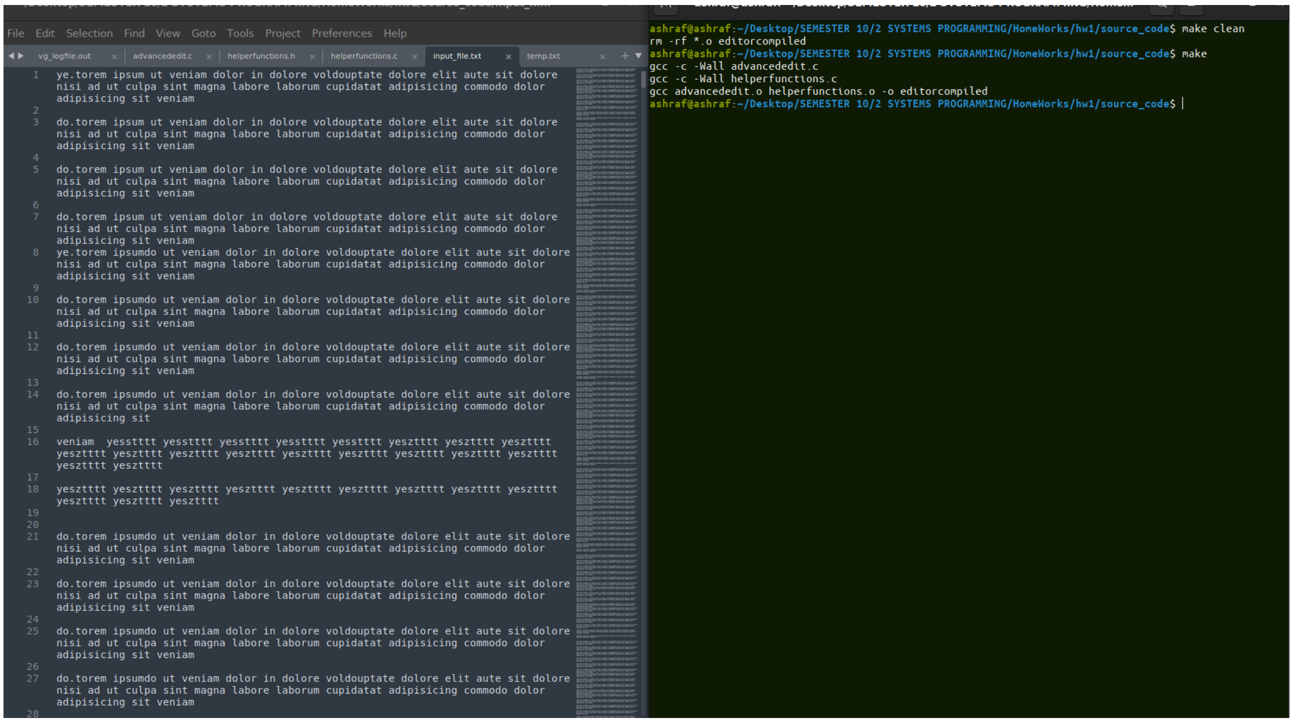
```c
char *to_lower_string(char *str){
   unsigned char *pointer = (unsigned char *)str;
   while (*pointer) {
      *pointer = to_lower((unsigned char)*pointer);
      pointer++;
   }
   return str;
}

int to_lower(int character){ // custom to lower function

    if(character >= 'A' && character <= 'Z'){
        return (character + 'a' - 'A');

    }else{
        return(character);

    }

}
```

- In the above functions: to_lower_string() function takes a string inside it it calls the to_lower() function which lowers each given character.

- I have explicitly designed my program to not to get any warning and memory leak error.

Below is the test screen shot of the program:

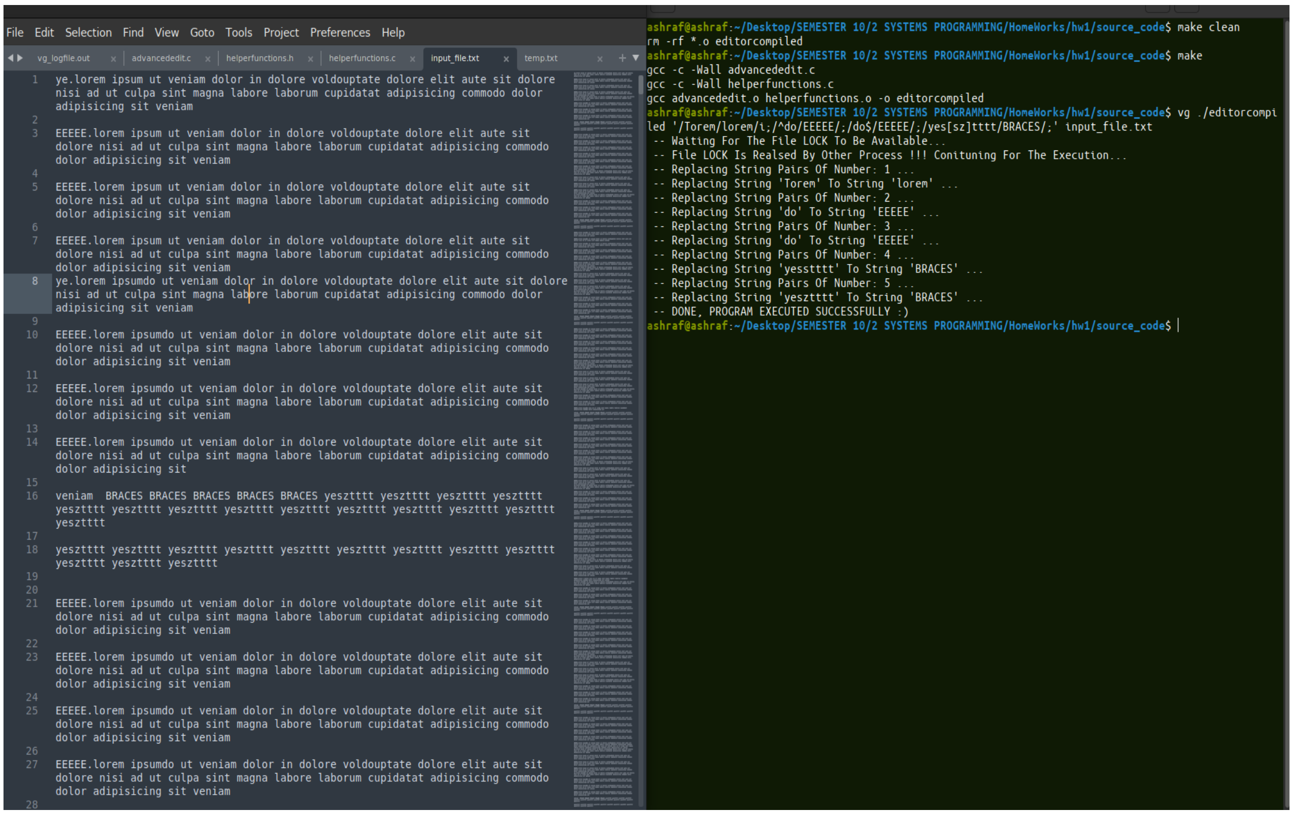
BEFORE REPLACEMENT:



```
ye.torem ipsum ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsum ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsum ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsum ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam
ye.torem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit

veniam  yessttt yessttt yessttt yessttt yessttt yeszttt yeszttt yeszttt
yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt
yeszttt yeszttt

yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt
yeszttt yeszttt yeszttt


do.torem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

do.torem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam
```

```
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw1/source_code$ make clean
rm -rf *.o editorcompiled
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw1/source_code$ make
gcc -c -Wall advancededit.c
gcc -c -Wall helperfunctions.c
gcc advancededit.o helperfunctions.o -o editorcompiled
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw1/source_code$ |
```

AFTER REPLACEMENT:



```
ye.lorem ipsum ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

EEEEE.lorem ipsum ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam

EEEEE.lorem ipsum ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam

EEEEE.lorem ipsum ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam
ye.lorem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit dolore
nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo dolor
adipisicing sit veniam

EEEEE.lorem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam

EEEEE.lorem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam

EEEEE.lorem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit

veniam  BRACES BRACES BRACES BRACES BRACES yeszttt yeszttt yeszttt yeszttt
yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt
yeszttt

yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt yeszttt
yeszttt yeszttt yeszttt


EEEEE.lorem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam

EEEEE.lorem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam

EEEEE.lorem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam

EEEEE.lorem ipsumdo ut veniam dolor in dolore voldouptate dolore elit aute sit
dolore nisi ad ut culpa sint magna labore laborum cupidatat adipisicing commodo
dolor adipisicing sit veniam
```

```
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw1/source_code$ make clean
rm -rf *.o editorcompiled
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw1/source_code$ make
gcc -c -Wall advancededit.c
gcc -c -Wall helperfunctions.c
gcc advancededit.o helperfunctions.o -o editorcompiled
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw1/source_code$ vg ./editorcompi
led '/Torem/lorem/i;/^do/EEEEE/;/do$/EEEEE/;/yes[sz]tttt/BRACES/;' input_file.txt
 -- Waiting For The File LOCK To Be Available...
 -- File LOCK Is Realsed By Other Process !!! Conituning For The Execution...
 -- Replacing String Pairs Of Number: 1 ...
 -- Replacing String 'Torem' To String 'lorem' ...
 -- Replacing String Pairs Of Number: 2 ...
 -- Replacing String 'do' To String 'EEEEE' ...
 -- Replacing String Pairs Of Number: 3 ...
 -- Replacing String 'do' To String 'EEEEE' ...
 -- Replacing String Pairs Of Number: 4 ...
 -- Replacing String 'yessttt' To String 'BRACES' ...
 -- Replacing String Pairs Of Number: 5 ...
 -- Replacing String 'yeszttt' To String 'BRACES' ...
 -- DONE, PROGRAM EXECUTED SUCCESSFULLY :)
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw1/source_code$ |
```