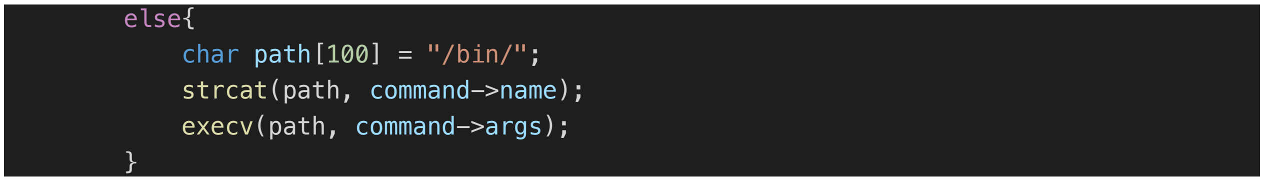
Furkan Yakal 59943

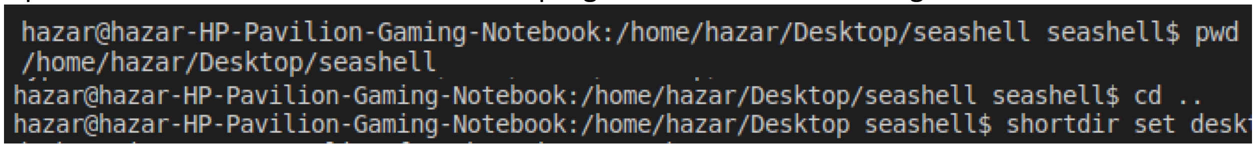
Comp 304
Project1 Report

Question 1. (Completed)
As stated in the question 1 description, the execv() method requires the path and argument list as inputs. The common linux programs can be executed under the /bin folder. Therefore, the following additions are made for the question 1.



```
        else{
            char path[100] = "/bin/";
            strcat(path, command->name);
            execv(path, command->args);
        }
```

Upon this addition all the common Linux programs are started working.

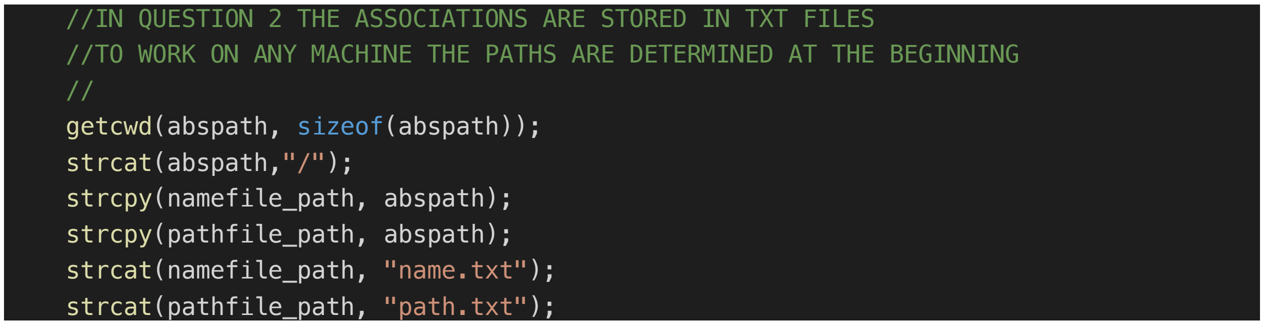


```
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ pwd
/home/hazar/Desktop/seashell
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ cd ..
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop seashell$ shortdir set desk
```

Question 2. (Completed)
In the second question, the shortdir command is introduced with 5 different supportive options (set, jump, del, clear, list).

The name-path associations are stored in the two different .txt files. “name.txt and “path.txt”. The association’s line numbers are the same with each other. As a result of that any deletion or updates can be done with ease.

To save associations between different sessions the .txt files are used. And the paths of these files are determined in the very beginning of the program with the same folder as the seashell.c file.



```
    //IN QUESTION 2 THE ASSOCIATIONS ARE STORED IN TXT FILES
    //TO WORK ON ANY MACHINE THE PATHS ARE DETERMINED AT THE BEGINNING
    //
    getcwd(abspath, sizeof(abspath));
    strcat(abspath,"/");
    strcpy(namefile_path, abspath);
    strcpy(pathfile_path, abspath);
    strcat(namefile_path, "name.txt");
    strcat(pathfile_path, "path.txt");
```

For the sake of simplicity, the procedures are completed inside the helper methods.

```
else if (strcmp(command->name, "shortdir") == 0)
    {
        if(strcmp(command->args[0], "set") == 0 && command->args[1]!= NULL){
            char cwd[500];
            set_name(command->args[1],getcwd(cwd, sizeof(cwd)));
        }else if(strcmp(command->args[0], "jump") == 0 && command->args[1]!= NULL){
            char *path = get_path(command->args[1]);
            if(strcmp(path, " ")!= 0){
                chdir(path);
            }
        }else if(strcmp(command->args[0], "del") == 0 && command->args[1]!= NULL){
            delete(command->args[1]);
        }else if(strcmp(command->args[0], "clear") == 0 && command->args[1]== NULL){
            clear();
        }else if(strcmp(command->args[0], "list") == 0 && command->args[1]== NULL){
            list_associations();
        }else{
            printf("Wrong argument %s\n", command->args[0]);
        }

        return SUCCESS;
    }
```

```
//METHODS USED IN Q2
int find_index(char* str1, char* fileName);
void clear();
void update_txt(int line_number, char *new_txt, char *file_name);
void set_name(char *name, char *path);
void delete(char *name);
void list_associations();
char* get_path(char *name);
```

Set:
Firstly, controls the name.txt and path.txt files for checking possible previously defined associations. If there is any association exists, these lines are removed from both .txt files. At the end the given associations are added to the corresponding .txt files.

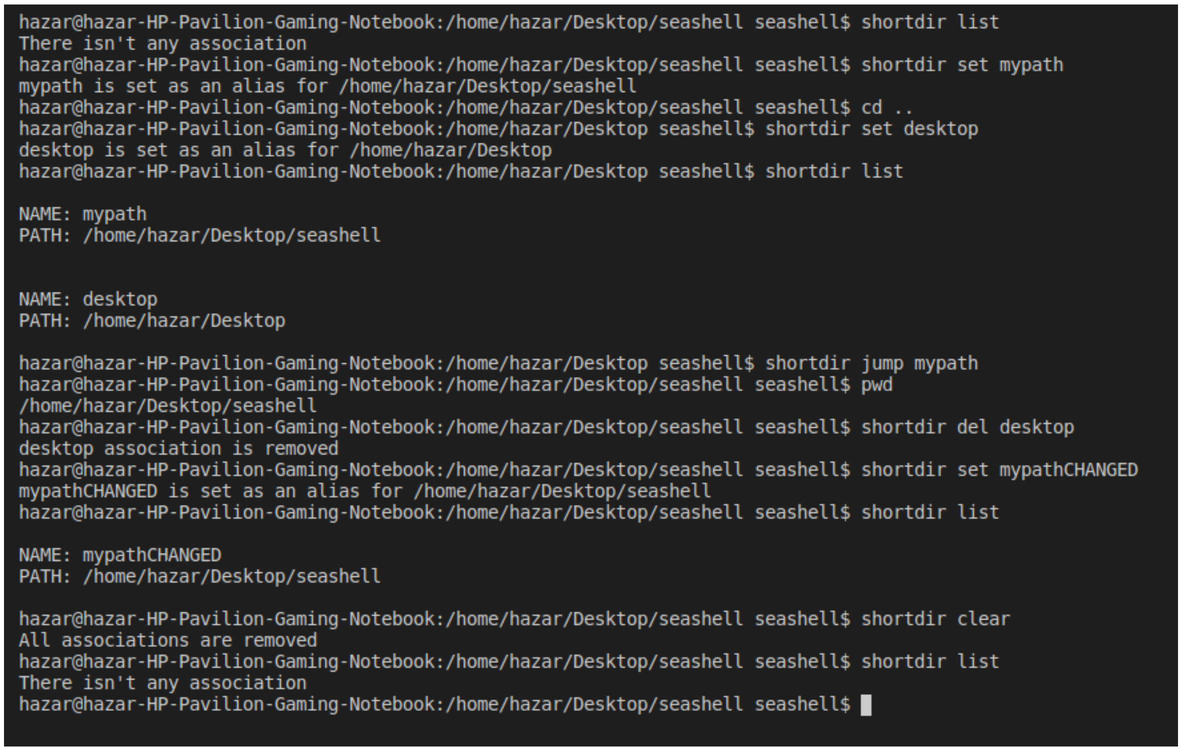
Jump:
With the given name, finds the line number in the “name.txt” file then, gets the corresponding path from the path.txt file. Calls the chdir method for changing the directory. If the name association does not exist displays a warning in the console.

Del:

With the given name, finds the line number in the “name.txt” file then, deletes the corresponding lines from “name.txt” and “path.txt” files.

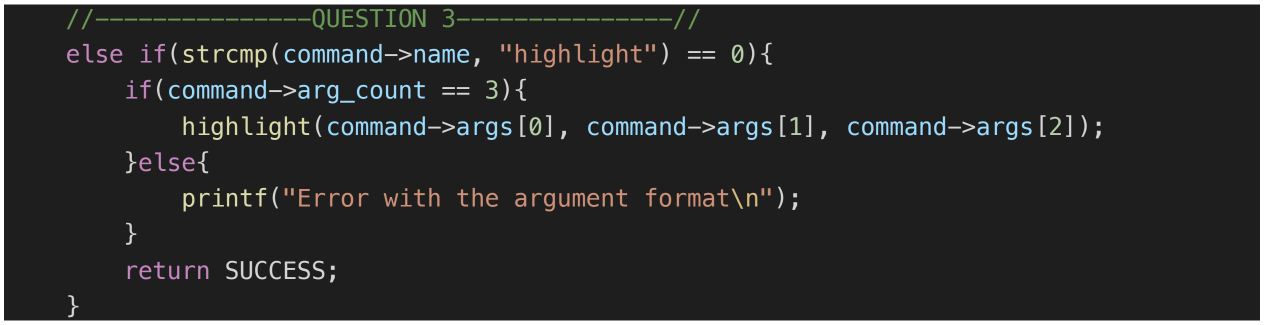
Clear:
Deletes the “name.txt” and “path.txt” files

List:
Displays all the names and their corresponding paths by reading the .txt files.

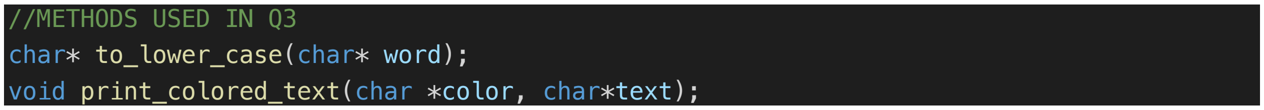


```
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ shortdir list
There isn't any association
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ shortdir set mypath
mypath is set as an alias for /home/hazar/Desktop/seashell
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ cd ..
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop seashell$ shortdir set desktop
desktop is set as an alias for /home/hazar/Desktop
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop seashell$ shortdir list

NAME: mypath
PATH: /home/hazar/Desktop/seashell


NAME: desktop
PATH: /home/hazar/Desktop

hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop seashell$ shortdir jump mypath
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ pwd
/home/hazar/Desktop/seashell
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ shortdir del desktop
desktop association is removed
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ shortdir set mypathCHANGED
mypathCHANGED is set as an alias for /home/hazar/Desktop/seashell
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ shortdir list

NAME: mypathCHANGED
PATH: /home/hazar/Desktop/seashell

hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ shortdir clear
All associations are removed
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ shortdir list
There isn't any association
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ █
```

Question 3. (Completed)
Given the word, color option and the .txt file. All the occurrences of the word is changed with the given color option.



```
    //---------------QUESTION 3---------------//
    else if(strcmp(command->name, "highlight") == 0){
        if(command->arg_count == 3){
            highlight(command->args[0], command->args[1], command->args[2]);
        }else{
            printf("Error with the argument format\n");
        }
        return SUCCESS;
    }
```



```
//METHODS USED IN Q3
char* to_lower_case(char* word);
void print_colored_text(char *color, char*text);
```

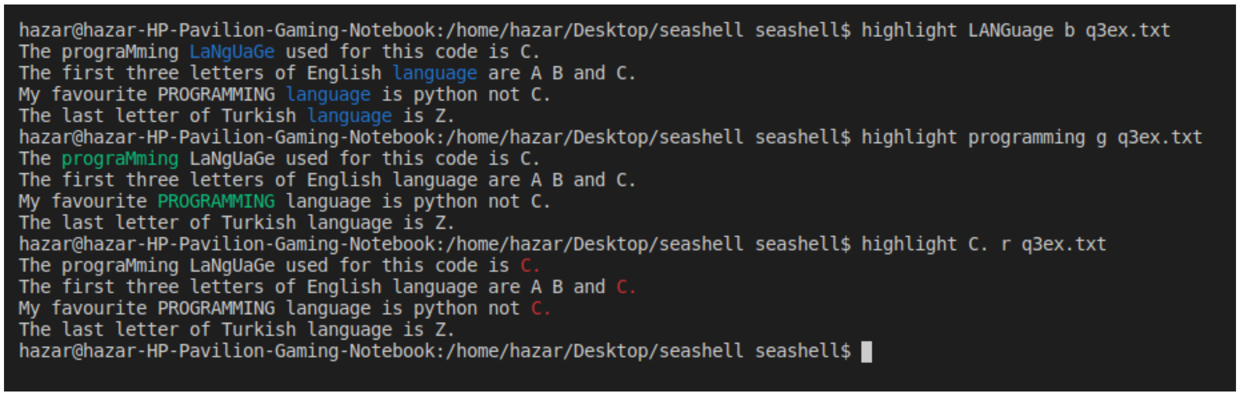
```c
void highlight(char *word, char* color, char* file_name);
```

The one crucial point is that the word check is case insensitive. So, the given word, and the every word in the .txt file is converted to lowercase version.

```c
//since the search is case-insensitive all the words are replaced with lower versions
char* to_lower_case(char* word){
   char *new_word = malloc(100);
   strcpy(new_word, word);

   for(int i=0;i<=strlen(new_word);i++){
      if(new_word[i]>=65&&new_word[i]<=90){
         new_word[i]=new_word[i]+32;
      }
   }
   return new_word;
}
```

If the words are matched, then it is printed with the following method.

```c
//displaying colored text r, g or b
void print_colored_text(char *color, char*text){
   if (strcmp(color, "r") == 0){
      printf("\033[;31m%s \033[0m",text);
   }else if(strcmp(color, "b") == 0){
      printf("\033[;34m%s \033[0m",text);
   }else if(strcmp(color, "g") == 0){
      printf("\033[;32m%s \033[0m",text);
   }
}
```

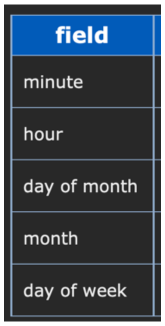


```
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ highlight LANGuage b q3ex.txt
The prograMming LaNgUaGe used for this code is C.
The first three letters of English language are A B and C.
My favourite PROGRAMMING language is python not C.
The last letter of Turkish language is Z.
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ highlight programming g q3ex.txt
The prograMming LaNgUaGe used for this code is C.
The first three letters of English language are A B and C.
My favourite PROGRAMMING language is python not C.
The last letter of Turkish language is Z.
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ highlight C. r q3ex.txt
The prograMming LaNgUaGe used for this code is C.
The first three letters of English language are A B and C.
My favourite PROGRAMMING language is python not C.
The last letter of Turkish language is Z.
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ 
```

Question 4. (Completed)

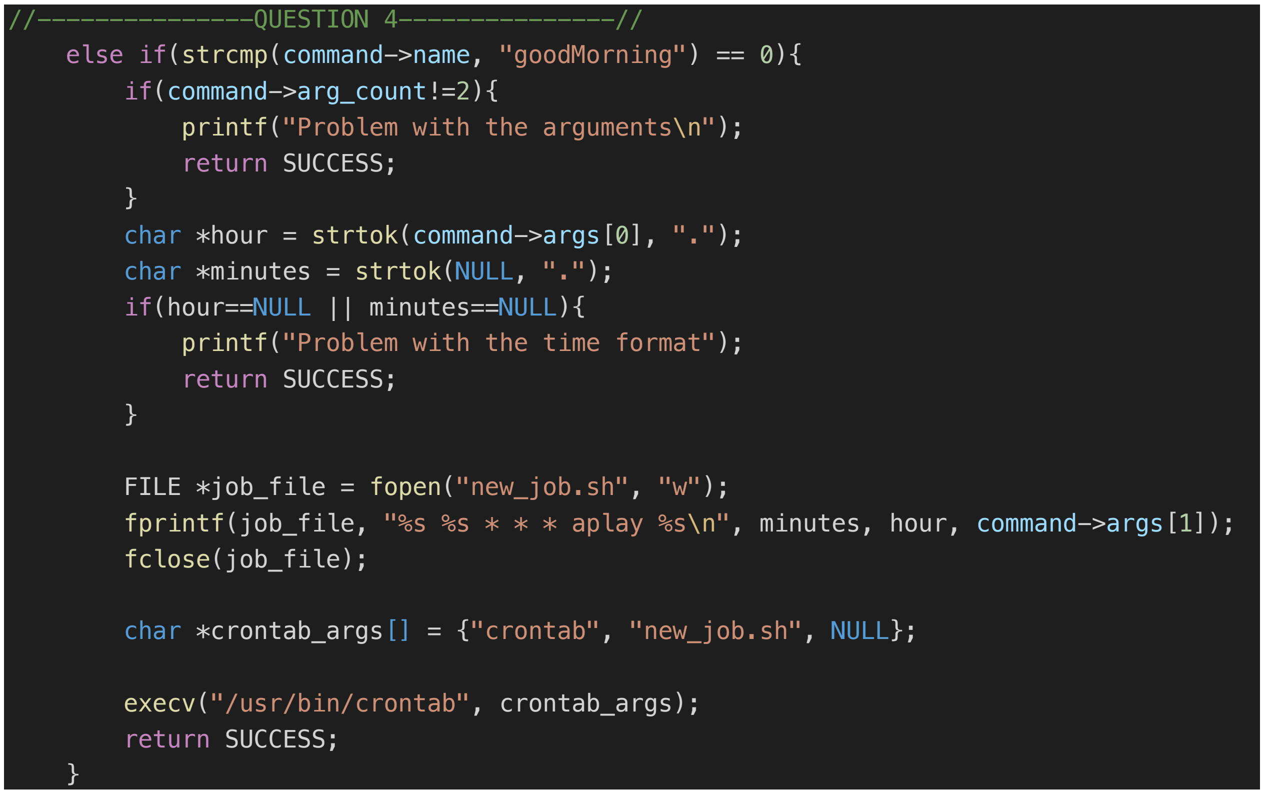
The goodMorning command is defined which takes time argument in “hh.mm” format and a music file argument.

The crontab takes several scheduling time inputs. In our scenario, only the hour and minute parameters are needed. So, with the strtok function the hour and minute parameters are stored in corresponding variables.

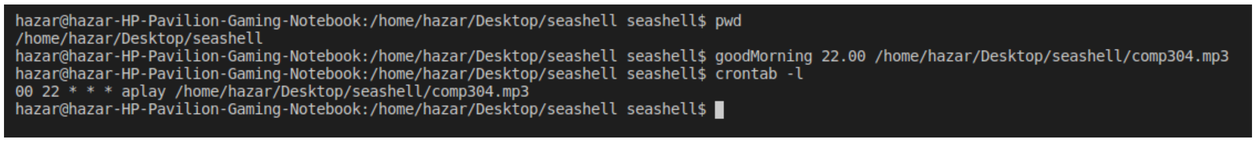


field
minute
hour
day of month
month
day of week

Then in the created sh file, the time parameters and the song are given.
As discussed in the first question, the execv requires the exact path in this case it is /usr/bin/crontab.



```
//---------------QUESTION 4---------------//
    else if(strcmp(command->name, "goodMorning") == 0){
        if(command->arg_count!=2){
            printf("Problem with the arguments\n");
            return SUCCESS;
        }
        char *hour = strtok(command->args[0], ".");
        char *minutes = strtok(NULL, ".");
        if(hour==NULL || minutes==NULL){
            printf("Problem with the time format");
            return SUCCESS;
        }

        FILE *job_file = fopen("new_job.sh", "w");
        fprintf(job_file, "%s %s * * * aplay %s\n", minutes, hour, command->args[1]);
        fclose(job_file);

        char *crontab_args[] = {"crontab", "new_job.sh", NULL};

        execv("/usr/bin/crontab", crontab_args);
        return SUCCESS;
    }
```



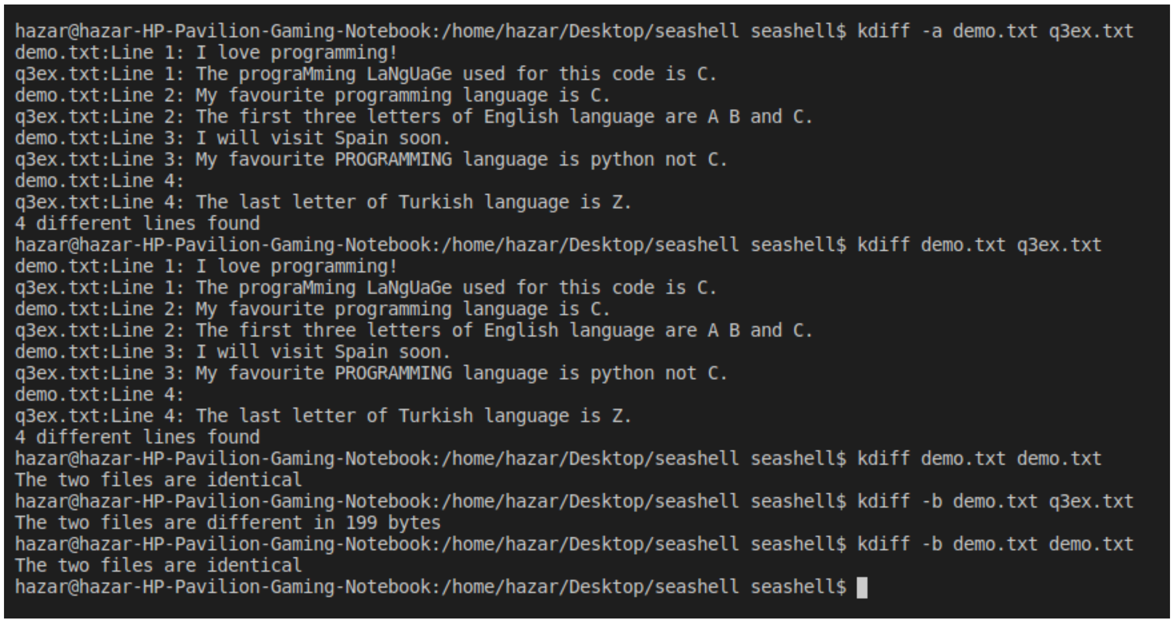
```
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ pwd
/home/hazar/Desktop/seashell
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ goodMorning 22.00 /home/hazar/Desktop/seashell/comp304.mp3
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ crontab -l
00 22 * * * aplay /home/hazar/Desktop/seashell/comp304.mp3
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ 
```

Question 5. (Completed)
Kdiff command with 2 options,
-a Compares the two .txt files by different lines
-b Compares any file with any extension by bytes
If no option is given the default option will be -a.

-a
The two .txt files are compared until the longer one is finished. If the lines are different then the lines are displayed. If everything is the same then confirmation is displayed.
-b
By using the fread the characters are read in the binary form and each char is compared whether they are the same or not until the longer file finishes.

```
    else if(strcmp(command->name, "kdiff") == 0){
        if(command->arg_count<2){
            printf("Problem with the arguments\n");
        }
        else if(strcmp(command->args[0], "-b")==0){
            compare_binary_files(command->args[1], command->args[2]);
        }else{
            compare_txt_files(command->args[command->arg_count-2], command-
>args[command->arg_count-1]);
        }
        return SUCCESS;
    }
```

```
//METHODS USED IN Q5
void compare_txt_files(char *txt1, char *txt2);
void compare_binary_files(char *file1_name, char *file2_name);
```



```
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ kdiff -a demo.txt q3ex.txt
demo.txt:Line 1: I love programming!
q3ex.txt:Line 1: The prograMming LaNgUaGe used for this code is C.
demo.txt:Line 2: My favourite programming language is C.
q3ex.txt:Line 2: The first three letters of English language are A B and C.
demo.txt:Line 3: I will visit Spain soon.
q3ex.txt:Line 3: My favourite PROGRAMMING language is python not C.
demo.txt:Line 4:
q3ex.txt:Line 4: The last letter of Turkish language is Z.
4 different lines found
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ kdiff demo.txt q3ex.txt
demo.txt:Line 1: I love programming!
q3ex.txt:Line 1: The prograMming LaNgUaGe used for this code is C.
demo.txt:Line 2: My favourite programming language is C.
q3ex.txt:Line 2: The first three letters of English language are A B and C.
demo.txt:Line 3: I will visit Spain soon.
q3ex.txt:Line 3: My favourite PROGRAMMING language is python not C.
demo.txt:Line 4:
q3ex.txt:Line 4: The last letter of Turkish language is Z.
4 different lines found
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ kdiff demo.txt demo.txt
The two files are identical
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ kdiff -b demo.txt q3ex.txt
The two files are different in 199 bytes
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ kdiff -b demo.txt demo.txt
The two files are identical
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$
```

Question 6. (Completed)
In this question, the “concatenate” command is implemented. This command required at least 2 .txt arguments and can take as many .txt arguments as inputs. The first argument represents the name of the final .txt file. Starting from the second argument, the following .txt files are

consecutively concatenated under each other, and the final result is saved in the .txt file with the name determined from the first argument.

concatenate <input 1> ………………. <input n> where n>=2

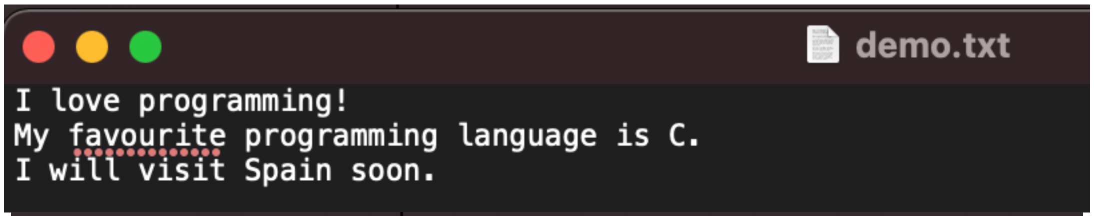
```
//---------------QUESTION 6---------------//
    else if(strcmp(command->name, "concatenate") == 0){
        concatenate_txt_files(command->arg_count, command->args);
        return SUCCESS;
    }
```

```

// QUESTION 6 HELPER METHOD START //

// # of arguments = 1....n where n>=2
// All arguments need to be in the .txt format.
// Starting from the second .txt file all .txt files are concatenated one by one.
// The resulting concatenated .txt file is saved with the name given in the first
argument.
void concatenate_txt_files(int argc, char *argv[]){
   if(argc<2){
      printf("Bad format there should be at least 2 arguments. %d is given\n", argc);
      return;
   }

   int l;
   int i;
   for (i=0; i<argc; i++){
      l = strlen(argv[i]);
      if(l<4 || argv[i][l-4]!='.' || argv[i][l-3]!='t' || argv[i][l-2]!='x' ||
argv[i][l-1]!='t'){
         printf("All files should be in txt format\n");
         return;
      }
   }

   FILE *output_txt, *input_txt;
   output_txt = fopen(argv[0], "a+");
   char line[256];

   for(i=1; i<argc;i++){
      input_txt = fopen(argv[i], "r");
      while(fgets(line, sizeof(line), input_txt)!=NULL){
         fputs(line, output_txt);
      }
      fclose(input_txt);
   }
```

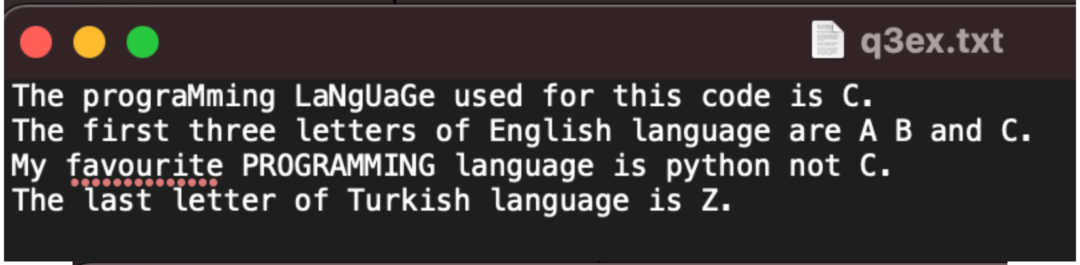
```
    fclose(output_txt);
}
```

As seen from the below line, many .txt files are given as input and the final result is stored in the myconcatenated.txt

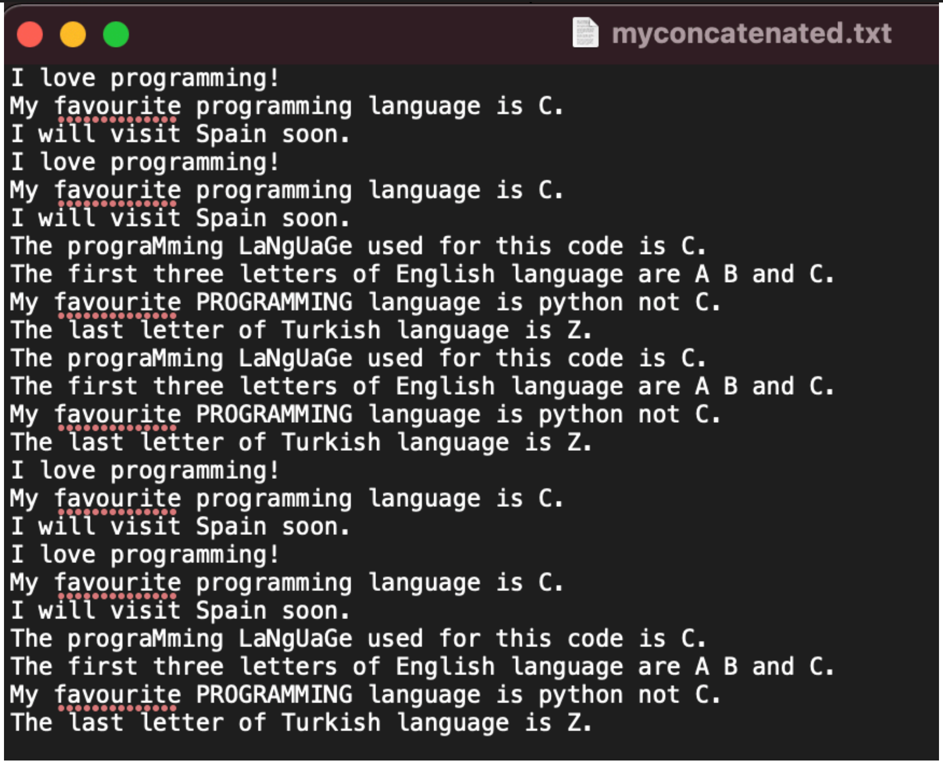
```
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$ concatenate myconcatenated.txt demo.txt demo.txt q3ex.txt q3ex.txt demo.txt demo.txt q3ex.txt
hazar@hazar-HP-Pavilion-Gaming-Notebook:/home/hazar/Desktop/seashell seashell$
```



```
I love programming!
My favourite programming language is C.
I will visit Spain soon.
```



```
The prograMming LaNgUaGe used for this code is C.
The first three letters of English language are A B and C.
My favourite PROGRAMMING language is python not C.
The last letter of Turkish language is Z.
```



```
I love programming!
My favourite programming language is C.
I will visit Spain soon.
I love programming!
My favourite programming language is C.
I will visit Spain soon.
The prograMming LaNgUaGe used for this code is C.
The first three letters of English language are A B and C.
My favourite PROGRAMMING language is python not C.
The last letter of Turkish language is Z.
The prograMming LaNgUaGe used for this code is C.
The first three letters of English language are A B and C.
My favourite PROGRAMMING language is python not C.
The last letter of Turkish language is Z.
I love programming!
My favourite programming language is C.
I will visit Spain soon.
I love programming!
My favourite programming language is C.
I will visit Spain soon.
The prograMming LaNgUaGe used for this code is C.
The first three letters of English language are A B and C.
My favourite PROGRAMMING language is python not C.
The last letter of Turkish language is Z.
```