MARMARA UNIVERSITY
FACULTY OF ENGINEERING

CSE 3033 ASSIGNMENT #1 REPORT

Name, Surname: Yasin Çörekci - Ertan Karaoğlu - Busenur Yılmaz

Student ID: 150119858– 150119066 – 150119683
Department: Computer Engineering

Table Of Contents:

Project3.h File

ts_Queue;

This struct is used to represent a singly-linked list that contains lines read from the input file, where each node of the list holds information about a line: the line number, the starting position and ending position, the line content, and some flags indicating whether the line has been converted to uppercase, replaced or written to the output file.

```c
typedef struct
{
    struct ts_Queue *previusLine;
    int line;
    long start_pos;
    long end_pos;
    char *msg;
    int isUpper_done;
    int isReplace_done;
    int isWrite_done;
    pthread_mutex_t mutex;
    struct ts_Queue *nextLine;
}ts_Queue;
```

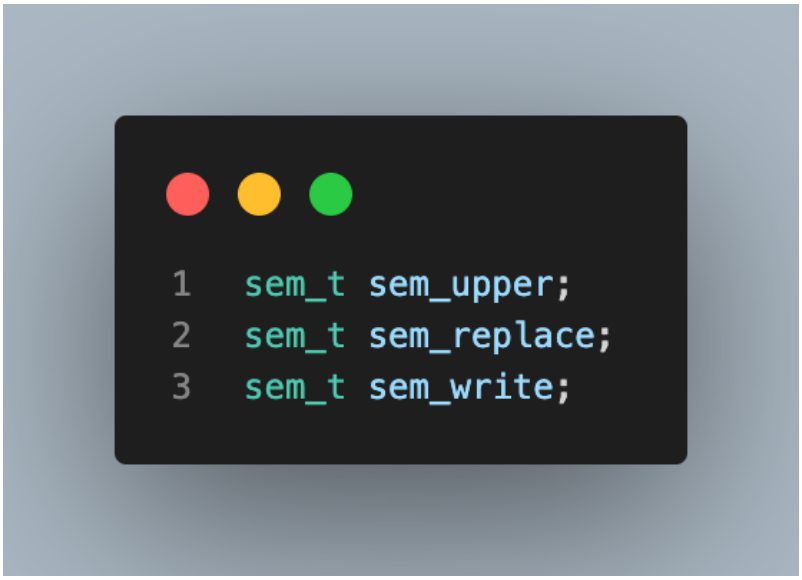
te_Jobs;

This typedef enum creates an enum for jobs.

```c
typedef enum
{
    upperJob,
    replaceJob,
    writeJob
}te_Jobs;
```

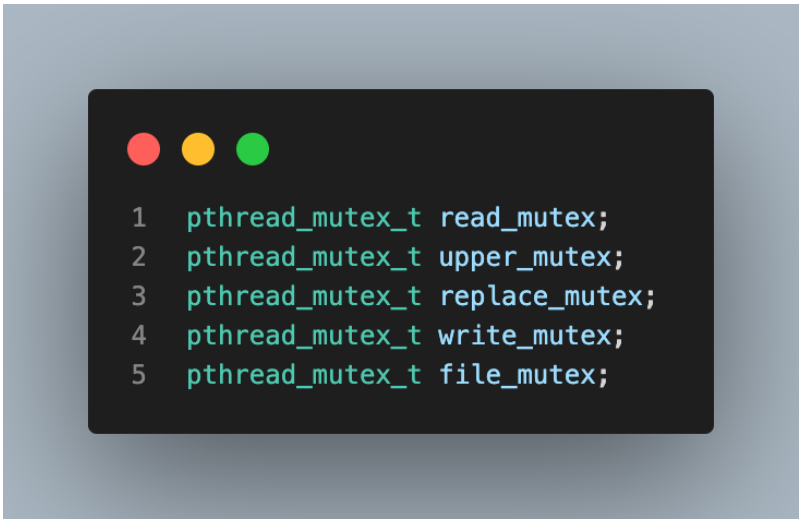
Semaphores;

We are using 3 semaphores; sem_upper, sem_replace, sem_write. When read threads read a line it post a signal to sem_upper and sem_replace. When upper and replace threads done they post signal to sem_write and this way write threads can start working.



Mutexes;

We have 5 mutex; read_mutex, upper_mutex, replace_mutex. write_mutex, file_mutex. While mutexes other than file_mutex ensure that each thread performs one operation at the same time and there is no synchronization problem, the file_mutex prevents confusion between read and write threads during read-write from the file.



Project3.c File

addLine

The function creates a new node and sets its fields, such as line number, message, with the given values. It also sets the flags to false indicating whether the row has been processed. Then it checks if the linked list is empty, if it is it sets the new node as the head of the list. If not, it finds the last node of the list and places the next in its node

```c
ts_Queue* addLine(int newLine,char *line)
{
    ts_Queue* newNode = (ts_Queue*)malloc(sizeof(ts_Queue));
    newNode->line = newLine;
    newNode->msg = (char*)malloc(strlen(line) + 1);
    strcpy(newNode->msg, line);
    pthread_mutex_init(&newNode->mutex, NULL);
    ts_Queue *currentNode = Header;
    if(Header==NULL)
    {
        Header=newNode;
        return newNode;
    }
    while (currentNode->nextLine != NULL)
    {
        currentNode = (ts_Queue*)currentNode->nextLine;

    }
    currentNode->nextLine = (struct ts_Queue*) newNode;
    newNode->previusLine=(struct ts_Queue*) currentNode;

    return newNode;
}
```

getNextAvailable

This function takes a job as an input and travels all nodes starting from the beginning of the queue and returns the first unexecuted node it sees.If there is no node the execute return NULL.

```c
ts_Queue* getNextAvailableLine(int job)
{
    for(;;)
    {
        ts_Queue* currentNode=Header;
        while(currentNode!=NULL)
        {
            if(pthread_mutex_trylock(&currentNode->mutex)==0)
            {

                if(job==upperJob && currentNode->isUpper_done==0)
                {
                    return currentNode;
                }
                else if(job==replaceJob && currentNode->isReplace_done==0)
                {
                    return currentNode;
                }
                else if(job==writeJob && currentNode->isWrite_done==0 && currentNode->isReplace_done==1 && currentNode->isUpper_done==1)
                {
                    return currentNode;
                }
                else
                {
                    pthread_mutex_unlock(&currentNode->mutex);
                }
            }
            currentNode=(ts_Queue*)currentNode->nextLine;
        }
        if(threadAllDone()==1)
        {
            return NULL;
        }
    }
}
```

threadAllDone

It checks that all flags of all nodes are done. If all are done, it returns 1, if not, it returns 0.

```
int threadAllDone()
{
    ts_Queue* currentNode=Header;
    int temp=0;
    while(currentNode!=NULL)
    {
        if(currentNode->isUpper_done==1 && currentNode->isReplace_done==1 && currentNode->isWrite_done==1)
        {
        }
        else
        {
            return 0;
        }
        currentNode=(ts_Queue*)currentNode->nextLine;
    }
    return 1;
}
```

readThreadFunc

This function takes thread_id as argument. It stays in an unlimited loop until the file it reads is finished and reads the file line by line. When reading a line, it sends the start and end point of its pointer to the Node that returns from the addLine function. When the reading process of each line is finished, it posts the semaphore for upper and replace operations.

```
void *readThreadFunc(void *arg)
{
    long tid = (long)arg;
    int line_num;
    char line[MAX_LINE_LENGTH];
    char temp[MAX_LINE_LENGTH];
    int size;
    for(;;)
    {
        memset(temp, 0, sizeof(temp));
        pthread_mutex_lock(&read_mutex);
        line_num = num_lines++;
        pthread_mutex_lock(&write_mutex);
        long start_pos= ftell(file);
        if(fgets(line, MAX_LINE_LENGTH, file)!=NULL)
        {
            long end_pos = ftell(file);
            pthread_mutex_unlock(&read_mutex);
            pthread_mutex_unlock(&write_mutex);
            for(int i = 0; i < strlen(line); i++)
            {
                if(line[i] != '\n')
                {
                    temp[i] = line[i];
                }
            }
            ts_Queue* currentNode = addLine(line_num, temp);
            currentNode->end_pos=end_pos;
            currentNode->start_pos=start_pos;
            printf("Read_%ld  \t\tRead_%ld read the line %d which is %s \n",tid,tid,line_num,temp);
            sem_post(&sem_upper);//Say to upper_func i read a line and you can upper a line
            sem_post(&sem_replace);//Say to replace_func i read a line and you can upper a line
            usleep(10);
        }
        else
        {
            pthread_mutex_unlock(&write_mutex);
            pthread_mutex_unlock(&read_mutex);
            sem_post(&sem_upper);
            sem_post(&sem_replace);
            num_lines--;
            return NULL;
        }
    }
}
```

replaceThreadFunc

This function stays in wait state until the read function posts the semaphore, when a line is read it enters the race condition with the replace function and tries to get the next node in the queue. After receiving it, it loops through all the letters of the message in the node one by one and replaces the spaces with _. At the end of the program, if the upper flag of this node is ==1, it will post the write semaphore.

```c
void *replaceThreadFunc(void *arg)
{
    long tid = (long)arg;
    char temp[MAX_LINE_LENGTH];
    for(;;)
    {
        sem_wait(&sem_replace);
        pthread_mutex_lock(&replace_mutex);
        ts_Queue* currentNode = getNextAvailableLine(replaceJob);
        pthread_mutex_unlock(&replace_mutex);
        if(currentNode==NULL)
        {
            sem_post(&sem_write);
            sem_post(&sem_replace);
            return NULL;
        }
        strcpy(temp,currentNode->msg);
        for(int i = 0; i < strlen(temp); i++)
        {
            if(temp[i]==' ')
            {
                currentNode->msg[i]='_';
            }
        }
        printf("Replace_%ld\t\tReplace_%ld read index %d and converted “%s” to “%s”\n",tid,tid,currentNode->line,temp,currentNode->msg);
        currentNode->isReplace_done=1;
        if(currentNode->isUpper_done==1)
        {
            sem_post(&sem_write);
        }
        pthread_mutex_unlock(&currentNode->mutex);
    }

    return NULL;
}
```

upperThreadFunc

This function stays in wait state until read function posts semaphore, when a line is read it enters race condition with upper function and tries to get next node in queue. After receiving it, it loops through all the letters of the message in the node one by one and rewrites it with capital letters. At the end of the program, if the replace flag of this node ==1, it posts the write semaphore.

```c
void *upperThreadFunc(void *arg)
{
    long tid = (long)arg;
    char temp[MAX_LINE_LENGTH];
    for(;;)
    {
        sem_wait(&sem_upper);
        pthread_mutex_lock(&upper_mutex);
        ts_Queue* currentNode = getNextAvailableLine(upperJob);
        pthread_mutex_unlock(&upper_mutex);
        if(currentNode==NULL)
        {
            sem_post(&sem_write);
            sem_post(&sem_upper);
            return NULL;
        }
        strcpy(temp,currentNode->msg);
        if(currentNode->isUpper_done==0)
        {
            for(int i=0;i<strlen(temp);i++)
            {
                currentNode->msg[i]=toupper(temp[i]);
            }
            printf("Upper_%ld  \t\tUpper_%ld read index %d and converted “%s” to “%s”\n",tid,tid,currentNode->line,temp,currentNode->msg);
            currentNode->isUpper_done=1;
            if(currentNode->isReplace_done==1)
            {
                sem_post(&sem_write);
            }
        }
        pthread_mutex_unlock(&currentNode->mutex);
        usleep(10);
    }
    return NULL;
}
```

writeThreadFunc

This is the most important function among them. Because Read threads are currently reading the file using the pointer, it is difficult to print the message we want on the line we want. To solve this problem, We save the pointer of the read thread to a saved_pos, and then when We want to write a line, We place my pointer to start_pos value that We store in current node and write message to that line. If We leave it like this after receiving it, the read thread will continue to read from the last line We write, so We have to continue from the pointer position We saved, We assign the value of the pointer to the value We saved.

```c
void *writeThreadFunc(void *arg)
{
    long tid = (long)arg;
    int size;
    for(;;)
    {
        sem_wait(&sem_write);
        pthread_mutex_lock(&write_mutex);
        ts_Queue* currentNode = getNextAvailableLine(writeJob);
        pthread_mutex_unlock(&write_mutex);
        if(currentNode == NULL)//Tüm node'lar bittiyse
        {
            sem_post(&sem_write);
            return NULL;
        }
        printf("Write_%ld  \t\tWrite_%ld line %d back which “%s”\n",tid,tid,currentNode->line,currentNode->msg);
        long saved_pos = ftell(file);//Before changing the pointer of the file we store its position
        long start_pos =currentNode->start_pos;
        long end_pos =currentNode->end_pos;
        /*
        In this are we change the pointer of the file, we process it and restore it, during this process,
        we put a lock so that the read thread does not enter when the pointer is in a
        different place than the next line that the read thread will read.
        */
        pthread_mutex_lock(&file_mutex);
        fseek(file, start_pos, SEEK_SET);
        fprintf(file, "%s", currentNode->msg);
        fseek(file, saved_pos, SEEK_SET);
        pthread_mutex_unlock(&file_mutex);
        currentNode->isWrite_done = 1;
        pthread_mutex_unlock(&currentNode->mutex);
        usleep(10);
    }
    return NULL;
}
```

Main

Our first job inside the main is to parse the values we gave as arguments from the terminal and put them in the necessary places.

```c
if (argc != 8)
{
    printf("Wrong Inputs\n");
    return 1;
}
char fileName[32];
strcpy(fileName, argv[2]);
int nmbOfReadThreads = atoi(argv[4]);
int nmbOfUpperThreads = atoi(argv[5]);
int nmbOfReplaceThreads = atoi(argv[6]);
int nmbOfWriteThreads = atoi(argv[7]);
int nmbOfAllThreads=nmbOfReadThreads+nmbOfUpperThreads+nmbOfReplaceThreads+nmbOfWriteThreads;
```

Then init the semaphores, mutexes and open the file that we have named as an argument as read/write.

```c
file = fopen(fileName, "r+");
pthread_mutex_init(&read_mutex,NULL);
pthread_mutex_init(&upper_mutex,NULL);
pthread_mutex_init(&replace_mutex,NULL);
pthread_mutex_init(&file_mutex, NULL);
sem_init(&sem_upper,0,0);
sem_init(&sem_replace,0,0);
sem_init(&sem_write,0,0);
```

Then we create the required number of threads using the number of threads given as arguments and send them to the functions, and we send the thread_ids as arguments. When the threads are finished, they enter the join section.

```c
long i;
pthread_t threads[nmbOfAllThreads];
printf("<Thread-type and ID>\t<Output>\n");
for (i = 1; i <=nmbOfAllThreads; i++)
{
    if(i<=nmbOfReadThreads)
    {
        pthread_create(&threads[i], NULL, readThreadFunc, (void*)i);
    }
    else if(i>=nmbOfReadThreads && i<=nmbOfUpperThreads+nmbOfReadThreads)
    {
        pthread_create(&threads[i], NULL, upperThreadFunc, (void*)i-nmbOfReadThreads);
    }
    else if(i>=nmbOfReadThreads+nmbOfUpperThreads && i<=nmbOfUpperThreads+nmbOfReadThreads+nmbOfReplaceThreads)
    {
        pthread_create(&threads[i], NULL, replaceThreadFunc, (void*)i-nmbOfReadThreads-nmbOfUpperThreads);
    }
    else if(i>=nmbOfReadThreads+nmbOfUpperThreads+nmbOfReplaceThreads && i<=nmbOfUpperThreads+nmbOfReadThreads+nmbOfReplaceThreads+nmbOfWriteThreads)
    {
        pthread_create(&threads[i], NULL, writeThreadFunc, (void*)i-nmbOfReadThreads-nmbOfUpperThreads-nmbOfReplaceThreads);
    }
}

for (i = 1; i <= nmbOfAllThreads; i++)
{
    pthread_join(threads[i], NULL);
}
```

When the threads join, the txt is finished. We destroy the semaphore and mutexes and close the file and terminate the program

```c
pthread_mutex_destroy(&replace_mutex);
pthread_mutex_destroy(&read_mutex);
pthread_mutex_destroy(&upper_mutex);
pthread_mutex_destroy(&file_mutex);
sem_destroy(&sem_upper);
sem_destroy(&sem_replace);
sem_destroy(&sem_write);
fclose(file);
printf("Program Finished Succesfly\n");
// displayQueue();
return 0;
```

deneme.txt File

İnput:

```
0 opsys
1 yasin
2 deneme
3 merhaba
4 merhaba
5 test olayi
6 para para para
7 lorem
8 ipsum
9 lorem ipsum
10 adana
11 bursa
12 istanbul
13 ankara
14 gaziantap
15 lahana
16 elma
17 pirasa ve kavun
18 sanane
19 laylaylay
20 driving a car
21 flying an airplane
22 taking a train
23 riding a bus
24 cycling
25 diving in a submarine
26 flying in a helicopter
27 flying a kite
28 traveling
29 visiting tourist attractions
30 going for a nature walk
31 camping
32 fishing
33 swimming
34 diving
35 taking a vacation
36 sunbathing on the beach
37 beautiful weather
38 warm water
39 cooling off
40 beautiful views
41 beautiful days
42 swimming pool
43 surfing
44 water sports
45 beautiful memories
46 making new friends
47 fun activities
48 discovering new places
49 traveling
50 visiting historical places
51 cultural events
52 shopping
53 experiencing local flavors
54 discovering new foods
55 exploring local crafts
56 watching beautiful sunset views
57 exploring nature
58 climbing high mountains
59 exploring forests
60 seeing local animals
61 taking night walks
62 watching the stars
63 sitting around a campfire
64 singing songs
65 telling stories
66 playing guitar
67 listening to music
68 reading a book
69 writing
70 painting
71 taking photos
72 making videos
73 learning new skills
74 trying new things
75 discovering new hobbies
76 exploring new places
77 discovering local historical sites
78 exploring local arts events
79 visiting local museums
80 discovering local festivals
81 sona kalan dona kalir
```

Output:

```
0_OPSYS
1_YASIN
2_DENEME
3_MERHABA
4_MERHABA
5_TEST_OLAYI
6_PARA_PARA_PARA
7_LOREM
8_IPSUM
9_LOREM_IPSUM
10_ADANA
11_BURSA
12_ISTANBUL
13_ANKARA
14_GAZIANTAP
15_LAHANA
16_ELMA
17_PiRASA_VE_KAVUN
18_SANANE
19_LAYLAYLAY
20_DRIVING_A_CAR
21_FLYING_AN_AIRPLANE
22_TAKING_A_TRAIN
23_RIDING_A_BUS
24_CYCLING
25_DIVING_IN_A_SUBMARINE
26_FLYING_IN_A_HELICOPTER
27_FLYING_A_KITE
28_TRAVELING
29_VISITING_TOURIST_ATTRACTIONS
30_GOING_FOR_A_NATURE_WALK
31_CAMPING
32_FISHING
33_SWIMMING
34_DIVING
35_TAKING_A_VACATION
36_SUNBATHING_ON_THE_BEACH
37_BEAUTIFUL_WEATHER
38_WARM_WATER
39_COOLING_OFF
40_BEAUTIFUL_VIEWS
41_BEAUTIFUL_DAYS
42_SWIMMING_POOL
43_SURFING
44_WATER_SPORTS
45_BEAUTIFUL_MEMORIES
46_MAKING_NEW_FRIENDS
47_FUN_ACTIVITIES
48_DISCOVERING_NEW_PLACES
49_TRAVELING
50_VISITING_HISTORICAL_PLACES
51_CULTURAL_EVENTS
52_SHOPPING
53_EXPERIENCING_LOCAL_FLAVORS
54_DISCOVERING_NEW_FOODS
55_EXPLORING_LOCAL_CRAFTS
56_WATCHING_BEAUTIFUL_SUNSET_VIEWS
57_EXPLORING_NATURE
58_CLIMBING_HIGH_MOUNTAINS
59_EXPLORING_FORESTS
60_SEEING_LOCAL_ANIMALS
61_TAKING_NIGHT_WALKS
62_WATCHING_THE_STARS
63_SITTING_AROUND_A_CAMPFIRE
64_SINGING_SONGS
65_TELLING_STORIES
66_PLAYING_GUITAR
67_LISTENING_TO_MUSIC
68_READING_A_BOOK
69_WRITING
70_PAINTING
71_TAKING_PHOTOS
72_MAKING_VIDEOS
73_LEARNING_NEW_SKILLS
74_TRYING_NEW_THINGS
75_DISCOVERING_NEW_HOBBIES
76_EXPLORING_NEW_PLACES
77_DISCOVERING_LOCAL_HISTORICAL_SITES
78_EXPLORING_LOCAL_ARTS_EVENTS
79_VISITING_LOCAL_MUSEUMS
80_DISCOVERING_LOCAL_FESTIVALS
81_SONA_KALAN_DONA_KALIR
```

Example Run

```
ubuntu@primary:~/Home/Downloads/Project 3$ gcc Project3.c
ubuntu@primary:~/Home/Downloads/Project 3$ ./a.out -d deneme.txt -n 5 5 5 5
<Thread-type and ID>    <Output>
Read_5                  Read_5 read the line 0 which is 0 opsys
Upper_2                 Upper_2 read index 0 and converted “0 opsys” to “0 OPSYS”
Replace_1               Replace_1 read index 1 and converted “1 yasin” to “1_yasin”
Read_4                  Read_4 read the line 1 which is 1 yasin
Upper_3                 Upper_3 read index 1 and converted “1_yasin” to “1_YASIN”
Replace_2               Replace_2 read index 0 and converted “0 OPSYS” to “0_OPSYS”
Read_5                  Read_5 read the line 2 which is 2 deneme
Upper_1                 Upper_1 read index 2 and converted “2 deneme” to “2 DENEME”
Replace_3               Replace_3 read index 2 and converted “2 DENEME” to “2_DENEME”
Read_5                  Read_5 read the line 3 which is 3 merhaba
Upper_4                 Upper_4 read index 3 and converted “3 merhaba” to “3 MERHABA”
Replace_4               Replace_4 read index 3 and converted “3 MERHABA” to “3_MERHABA”
Write_1                 Write_1 line 0 back which “0_OPSYS”
Write_2                 Write_2 line 1 back which “1_YASIN”
Read_4                  Read_4 read the line 4 which is 4 merhaba
Upper_5                 Upper_5 read index 4 and converted “4 merhaba” to “4 MERHABA”
Replace_5               Replace_5 read index 4 and converted “4 MERHABA” to “4_MERHABA”
Read_5                  Read_5 read the line 5 which is 5 test olayi
Upper_2                 Upper_2 read index 5 and converted “5 test olayi” to “5 TEST OLAYI”
Write_3                 Write_3 line 2 back which “2_DENEME”
Read_5                  Read_5 read the line 6 which is 6 para para para
Upper_3                 Upper_3 read index 6 and converted “6 para para para” to “6 PARA PARA PARA”
Replace_2               Replace_2 read index 6 and converted “6 PARA PARA PARA” to “6_PARA_PARA_PARA”
Read_5                  Read_5 read the line 7 which is 7 lorem
Upper_1                 Upper_1 read index 7 and converted “7 lorem” to “7 LOREM”
Replace_3               Replace_3 read index 7 and converted “7 LOREM” to “7_LOREM”
Write_5                 Write_5 line 3 back which “3_MERHABA”
Write_4                 Write_4 line 4 back which “4_MERHABA”
Write_2                 Write_2 line 6 back which “6_PARA_PARA_PARA”
Write_1                 Write_1 line 7 back which “7_LOREM”
Replace_1               Replace_1 read index 5 and converted “5 TEST OLAYI” to “5_TEST_OLAYI”
Read_5                  Read_5 read the line 8 which is 8 ipsum
Upper_4                 Upper_4 read index 8 and converted “8 ipsum” to “8 IPSUM”
Replace_4               Replace_4 read index 8 and converted “8 IPSUM” to “8_IPSUM”
Read_3                  Read_3 read the line 9 which is 9 lorem ipsum
Upper_5                 Upper_5 read index 9 and converted “9 lorem ipsum” to “9 LOREM IPSUM”
Replace_5               Replace_5 read index 9 and converted “9 LOREM IPSUM” to “9_LOREM_IPSUM”
Read_4                  Read_4 read the line 10 which is 10 adana
Upper_2                 Upper_2 read index 10 and converted “10 adana” to “10 ADANA”
Replace_2               Replace_2 read index 10 and converted “10 ADANA” to “10_ADANA”
Write_2                 Write_2 line 8 back which “8_IPSUM”
Write_3                 Write_3 line 5 back which “5_TEST_OLAYI”
Read_4                  Read_4 read the line 11 which is 11 bursa
Upper_3                 Upper_3 read index 11 and converted “11 bursa” to “11 BURSA”
Replace_3               Replace_3 read index 11 and converted “11 BURSA” to “11_BURSA”
Read_4                  Read_4 read the line 12 which is 12 istanbul
Upper_1                 Upper_1 read index 12 and converted “12 istanbul” to “12 ISTANBUL”
Read_3                  Read_3 read the line 13 which is 13 ankara
Upper_4                 Upper_4 read index 13 and converted “13 ankara” to “13 ANKARA”
Replace_4               Replace_4 read index 12 and converted “12 ISTANBUL” to “12_ISTANBUL”
```