Registration Number: 19BCE2119

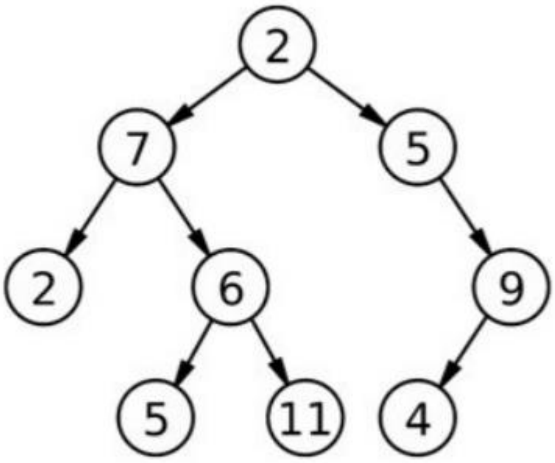
Name: Gaurav Kumar Singh

Final Assessment Test

Course: Operating Systems

QUESTION SET 1

1. A. Write a Multiprocess application where the parent process initiates the child process. The parent process creates the binary tree as like below.

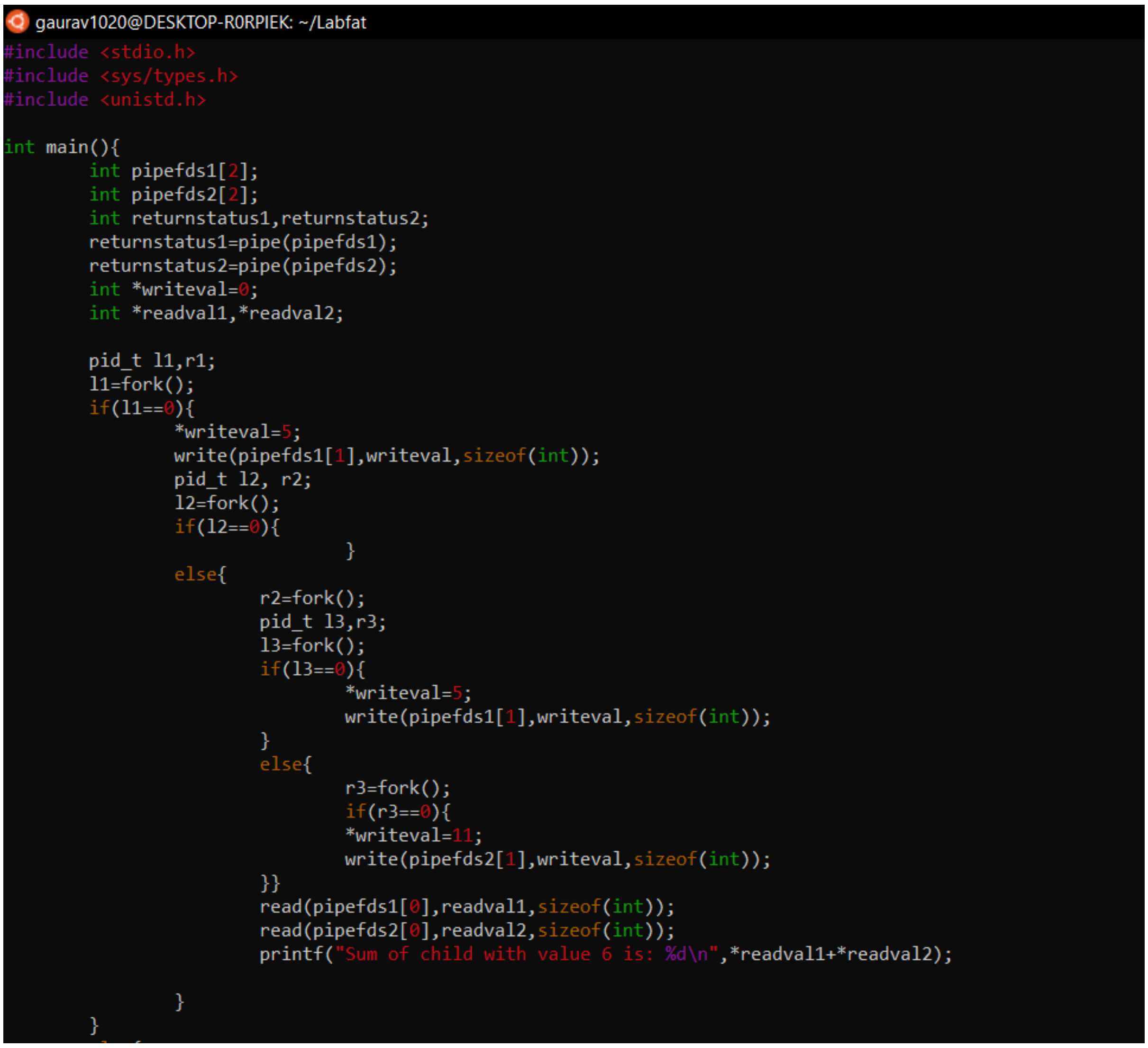


The child process does the task of finding the sum for the boundary nodes and returns the answer to the parent process. (10)

CODE:

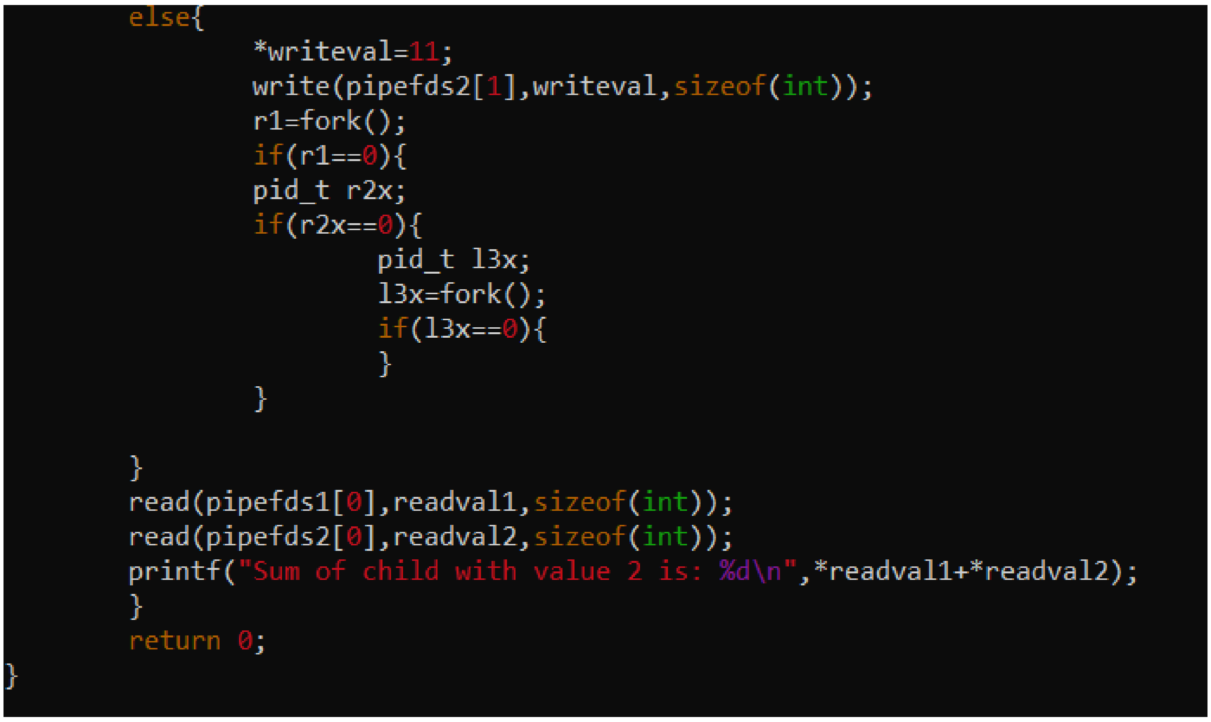
```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
	int pipefds1[2];
	int pipefds2[2];
	int returnstatus1,returnstatus2;
	returnstatus1=pipe(pipefds1);
	returnstatus2=pipe(pipefds2);
	int *writeval=0;
	int *readval1,*readval2;
```

```
        pid_t l1,r1;

        l1=fork();

        if(l1==0){

                *writeval=5;

                write(pipefds1[1],writeval,sizeof(int));

                pid_t l2, r2;

                l2=fork();

                if(l2==0){

                                }

                else{

                        r2=fork();

                        pid_t l3,r3;

                        l3=fork();

                        if(l3==0){

                                *writeval=5;

                                write(pipefds1[1],writeval,sizeof(int));

                        }

                        else{

                                r3=fork();

                                if(r3==0){

                                *writeval=11;

                                write(pipefds2[1],writeval,sizeof(int));

                        }}

                        read(pipefds1[0],readval1,sizeof(int));

                        read(pipefds2[0],readval2,sizeof(int));

                        printf("Sum of child with value 6 is: %d\n",*readval1+*readval2);


                }

        }

else{
        *writeval=11;

        write(pipefds2[1],writeval,sizeof(int));

        r1=fork();

        if(r1==0){

        pid_t r2x;

        if(r2x==0){

            pid_t l3x;

            l3x=fork();

            if(l3x==0){

            }

        }


    }

    read(pipefds1[0],readval1,sizeof(int));

    read(pipefds2[0],readval2,sizeof(int));

    printf("Sum of child with value 2 is: %d\n",*readval1+*readval2);

    }

    return 0;

}
```

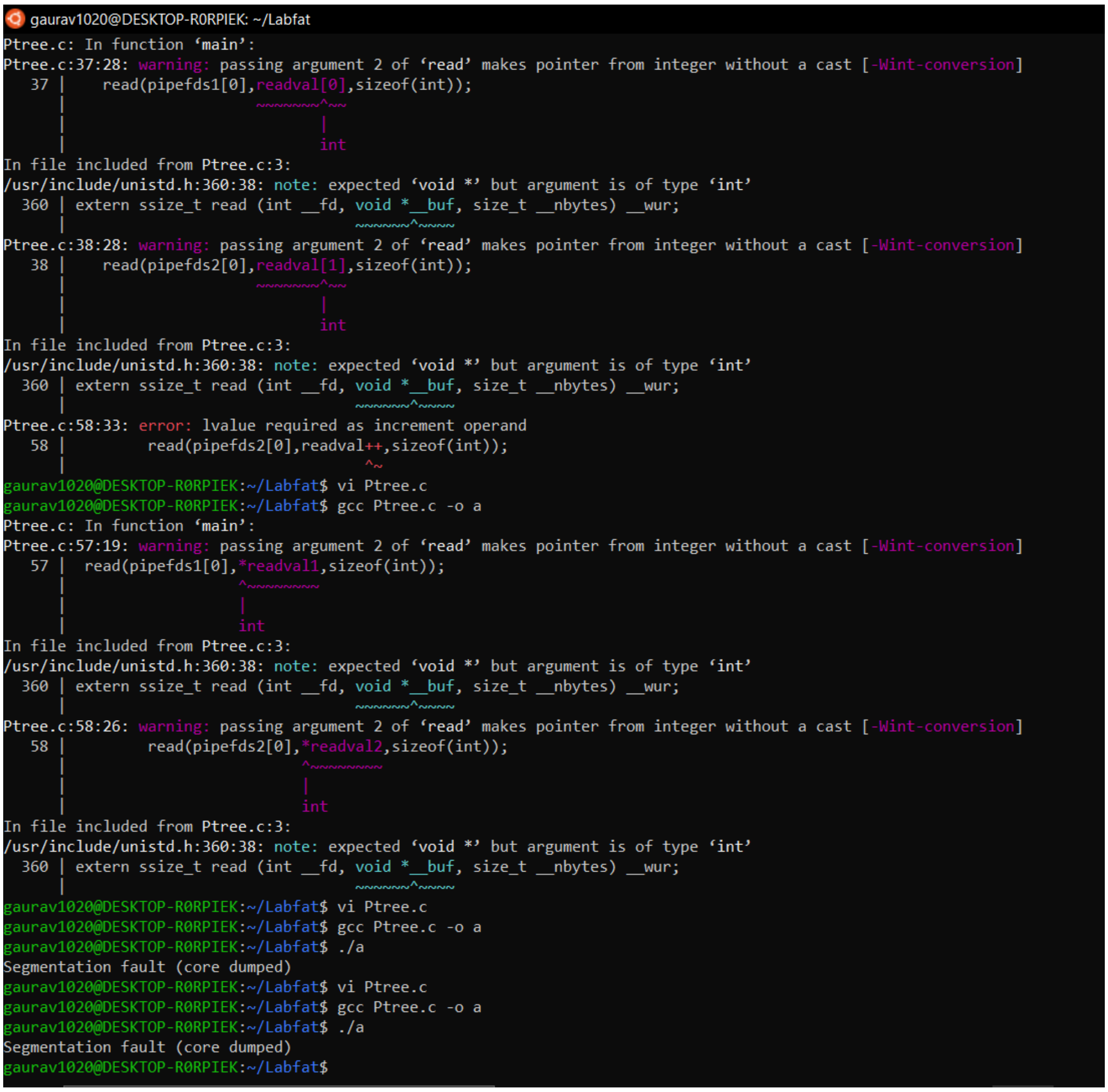


gaurav1020@DESKTOP-R0RPIEK: ~/Labfat

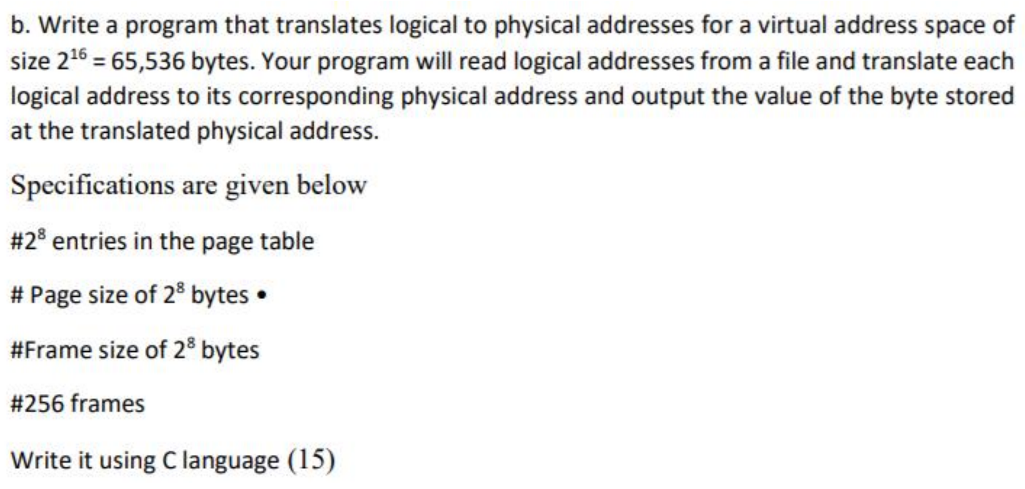
```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
        int pipefds1[2];
        int pipefds2[2];
        int returnstatus1,returnstatus2;
        returnstatus1=pipe(pipefds1);
        returnstatus2=pipe(pipefds2);
        int *writeval=0;
        int *readval1,*readval2;

        pid_t l1,r1;
        l1=fork();
        if(l1==0){
                *writeval=5;
                write(pipefds1[1],writeval,sizeof(int));
                pid_t l2, r2;
                l2=fork();
                if(l2==0){
                                }
                else{
                        r2=fork();
                        pid_t l3,r3;
                        l3=fork();
                        if(l3==0){
                                *writeval=5;
                                write(pipefds1[1],writeval,sizeof(int));
                        }
                        else{
                                r3=fork();
                                if(r3==0){
                                *writeval=11;
                                write(pipefds2[1],writeval,sizeof(int));
                        }}
                        read(pipefds1[0],readval1,sizeof(int));
                        read(pipefds2[0],readval2,sizeof(int));
                        printf("Sum of child with value 6 is: %d\n",*readval1+*readval2);

                }
        }
```



```c
        else{
                *writeval=11;
                write(pipefds2[1],writeval,sizeof(int));
                r1=fork();
                if(r1==0){
                pid_t r2x;
                if(r2x==0){
                        pid_t l3x;
                        l3x=fork();
                        if(l3x==0){
                        }
                }

        }
        read(pipefds1[0],readval1,sizeof(int));
        read(pipefds2[0],readval2,sizeof(int));
        printf("Sum of child with value 2 is: %d\n",*readval1+*readval2);
        }
        return 0;
}
```



```
gaurav1020@DESKTOP-R0RPIEK: ~/Labfat
Ptree.c: In function ‘main’:
Ptree.c:37:28: warning: passing argument 2 of ‘read’ makes pointer from integer without a cast [-Wint-conversion]
   37 |     read(pipefds1[0],readval[0],sizeof(int));
      |                      ~~~~~~~^~~
      |                             |
      |                             int
In file included from Ptree.c:3:
/usr/include/unistd.h:360:38: note: expected ‘void *’ but argument is of type ‘int’
  360 | extern ssize_t read (int __fd, void *__buf, size_t __nbytes) __wur;
      |                                ~~~~~~^~~~~
Ptree.c:38:28: warning: passing argument 2 of ‘read’ makes pointer from integer without a cast [-Wint-conversion]
   38 |     read(pipefds2[0],readval[1],sizeof(int));
      |                      ~~~~~~~^~~
      |                             |
      |                             int
In file included from Ptree.c:3:
/usr/include/unistd.h:360:38: note: expected ‘void *’ but argument is of type ‘int’
  360 | extern ssize_t read (int __fd, void *__buf, size_t __nbytes) __wur;
      |                                ~~~~~~^~~~~
Ptree.c:58:33: error: lvalue required as increment operand
   58 |          read(pipefds2[0],readval++,sizeof(int));
      |                                  ^~
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$ vi Ptree.c
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$ gcc Ptree.c -o a
Ptree.c: In function ‘main’:
Ptree.c:57:19: warning: passing argument 2 of ‘read’ makes pointer from integer without a cast [-Wint-conversion]
   57 |   read(pipefds1[0],*readval1,sizeof(int));
      |                    ^~~~~~~~~
      |                    |
      |                    int
In file included from Ptree.c:3:
/usr/include/unistd.h:360:38: note: expected ‘void *’ but argument is of type ‘int’
  360 | extern ssize_t read (int __fd, void *__buf, size_t __nbytes) __wur;
      |                                ~~~~~~^~~~~
Ptree.c:58:26: warning: passing argument 2 of ‘read’ makes pointer from integer without a cast [-Wint-conversion]
   58 |          read(pipefds2[0],*readval2,sizeof(int));
      |                           ^~~~~~~~~
      |                           |
      |                           int
In file included from Ptree.c:3:
/usr/include/unistd.h:360:38: note: expected ‘void *’ but argument is of type ‘int’
  360 | extern ssize_t read (int __fd, void *__buf, size_t __nbytes) __wur;
      |                                ~~~~~~^~~~~
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$ vi Ptree.c
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$ gcc Ptree.c -o a
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$ ./a
Segmentation fault (core dumped)
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$ vi Ptree.c
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$ gcc Ptree.c -o a
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$ ./a
Segmentation fault (core dumped)
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$
```



b. Write a program that translates logical to physical addresses for a virtual address space of size 2^{16} = 65,536 bytes. Your program will read logical addresses from a file and translate each logical address to its corresponding physical address and output the value of the byte stored at the translated physical address.

Specifications are given below

#2^8 entries in the page table

Page size of 2^8 bytes •

#Frame size of 2^8 bytes

#256 frames

Write it using C language (15)

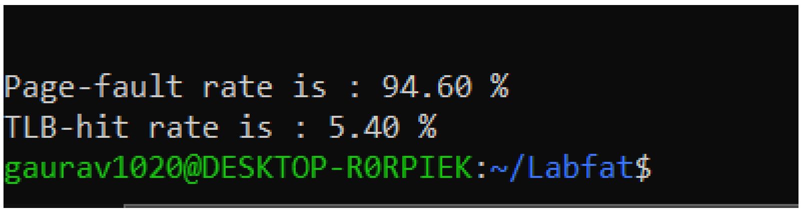
CODE

```
#include<stdio.h>


#include <stdlib.h>


int TLB[16][2];
int TLBsize = 0;


int pageTable[256];


int memoryFull = 0;
int memory[256][256];


void pageFault(int pageNum) {
  FILE * ptr = fopen("BACKING_STORE.bin", "rb");


  if (ptr == NULL) {
    printf("Couldn't open the file! \n");
    return;
```

```
 }

 if (fseek(ptr, 256 * pageNum, SEEK_SET) != 0) {
   printf("Page not found!\n");
   return;
 }

 unsigned char buffer[256];
 fread(buffer, sizeof(buffer), 1, ptr); // read 10 bytes to our buffer

 for (int i = 0; i < 256; i++) {
   memory[memoryFull][i] = buffer[i];
 }

 pageTable[pageNum] = memoryFull;
 memoryFull++;

 fclose(ptr);

}

int main(int argc, char * argv[]) {
 if (argc == 1)
   printf("No File name is passed in Command Line.\n");
 if (argc >= 2) {
   char * fileName = argv[1];
```

```
FILE * fp = fopen(fileName, "r");

if (fp == NULL) {
  printf("Couldn't open the file! \n");
  return -1;
}

int logicalAddress;
int pageFaultStat = 0, TLBstat = 0;
int totalAccess = 0;

for (int i = 0; i < 256; i++) {
  pageTable[i] = -1;
}

while (fscanf(fp, "%d", & logicalAddress) != EOF) {
  totalAccess++;

  int pageNumMask = 0, offsetMask = 0;
  for (int i = 0; i < 8; i++) {
    offsetMask |= 1 << i;
    pageNumMask |= 1 << (i + 8);
  }

  int pageNum = pageNumMask & logicalAddress;
  pageNum = pageNum >> 8;
```

```
    int offset = offsetMask & logicalAddress;

    int TLBhit = 0;

    for (int i = 0; i < TLBsize; i++) {
      if (TLB[i][0] == pageNum) {
        TLBhit = 1;
        int valueOfAdd = memory[TLB[i][1]][offset];
        int physicalAddress = TLB[i][1] * 256 + offset;
        printf("Virtual Address : %d and Physical Address : %d with Value : %d\n",
logicalAddress, physicalAddress, valueOfAdd);
        TLBstat++;
        break;
      }
    }

    if (TLBhit == 0) {
      if (pageTable[pageNum] == -1) {
        pageFault(pageNum);
        pageFaultStat++;
      }
      int valueOfAdd = memory[pageTable[pageNum]][offset];
      int physicalAddress = pageTable[pageNum] * 256 + offset;
      printf("Virtual Address : %d and Physical Address : %d with Value : %d \n",
logicalAddress, physicalAddress, valueOfAdd);

      if (TLBsize != 16) {
        TLB[TLBsize][0] = pageNum;
        TLB[TLBsize][1] = pageTable[pageNum];
        TLBsize++;
      } else {
        for (int i = 0; i < 15; i++) {
          TLB[i][0] = TLB[i + 1][0];
          TLB[i][1] = TLB[i + 1][1];
        }
        TLB[15][0] = pageNum;
        TLB[15][1] = pageTable[pageNum];
      }
    }
  }

  printf("\n\nPage-fault rate is : %.2f %%\n", (1.0 * pageFaultStat) / (1.0 *
totalAccess) * 100.0);
  printf("TLB-hit rate is : %.2f %%\n", (1.0 * TLBstat) / (1.0 * totalAccess) *
100.0);

  fclose(fp);
}
return 0;
}
```



```
Page-fault rate is : 94.60 %
TLB-hit rate is : 5.40 %
gaurav1020@DESKTOP-R0RPIEK:~/Labfat$
```