Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology, Pune-37

(Anautonomous Institute of Savitribai Phule Pune University)

Department of Computer Engineering

Group 8:

Division	**CS**
Batch	**B1**
Roll no.	**90**
Name	**Aditya Kurapati(Prn_no: 12320184)**

Phase 2:

Program Code:

Main.java

```
import java.io.*;
import java.util.Random;
class MyException extends Exception {
    public MyException(String s){
        super(s);
    }
}
public class Main{
    static BufferedReader fread;
    private  static FileReader fr;
    static int TLC,LLC,TTL,TLL,JobID,PI,SI,TI;
    static int j=0;
    static int PTR=0;
    FileWriter fw;

    static int memory_used;
    static boolean allocated[]=new boolean[30];
```

```
static boolean alloc[]=new boolean[30];

static int IC;
int random;
static int T;

static int PT[]=new int[40];

static String line;
static char [][]memory = new char[300][4];
static char []buffer=new char[40];
static char []IR=new char[4];
static char[] R =new char[4];

public void load(){



    Random r=new Random();
    try {
        fw = new FileWriter("output.txt");
        fr = new FileReader("input.txt");
```

```
        fread = new BufferedReader(fr);

        while ((line = fread.readLine()) != null) {

            buffer = line.toCharArray();

            if (buffer[0] == '$' && buffer[1] == 'A' && buffer[2] == 'M'
&& buffer[3] == 'J') {

                System.out.println("program card detected");
                init();
                pcb(buffer);



                continue;
            }
            if (buffer[0] == '$' && buffer[1] == 'D' && buffer[2] == 'T'
&& buffer[3] == 'A') {
                System.out.println("DATA card detected");
                executeUserProgram();
                continue;
```

```
            }
            if (buffer[0] == '$' && buffer[1] == 'E' && buffer[2] == 'N'
&& buffer[3] == 'D') {
                System.out.println("END card detected");
                System.out.println();
                print_memory();
                fw.write("\n");
                continue;

            }
            if (memory_used == 300) {
                System.out.println("Abort due to exceed memory usage");
            }

            for (int i = 0; i < line.length(); ) {

                {

                    int realAddress = addMap(memory_used);
                    memory[realAddress][i % 4] = buffer[i];
```

```
                i++;
                if(i%4==0) {

                    memory_used++;

                }
            }
        }
    }
    fw.close();
  }
  catch (Exception e){
    System.out.println(" Exception="+e);
  }

}
public int addMap(int value){
  if(allocated[value/10] == false){
    int num = allocate();
    int a = num%10;
    num = num/10;
```

```
    int b = num%10;
    num = num/10;
    int c = num%10;
    num = num/10;
    int d = num%10;
    allocated[value/10] = true;
    System.out.println(d+" "+c+" "+b+" "+a);
    memory[PTR+value/10][0] = (char)d;
    memory[PTR+value/10][1] = (char)c;
    memory[PTR+value/10][2] = (char)b;
    memory[PTR+value/10][3] = (char)a;
}

int I = (int)Math.floor(value/10);
if(I<0 || I>99)
    PI = 2;
int PTE = PTR + I;
int c = (int)(memory[PTE][2]);
int d = (int)(memory[PTE][3]);
int realAddress = Integer.parseInt(+c+""+d);
return (realAddress*10+(value%10));
```

```
    }
    private void pcb(char[] buffer) {
        JobID=
Integer.parseInt(String.valueOf(buffer[4]+""+buffer[5]+buffer[6]+buffer
[7]));

TTL=Integer.parseInt(String.valueOf(buffer[8]+""+buffer[9]+buffer[10]
+buffer[11]));

TLL=Integer.parseInt(String.valueOf(buffer[12]+""+buffer[13]+buffer[
14]+buffer[15]));
        System.out.println("Job ID = "+JobID);
        System.out.println("Time Limit= "+TTL);
        System.out.println("Line Limit= "+TLL);



    }



    public void init(){
        int page=allocate();
        memory_used=0;
```

```
    memory=new char[300][4];
    T=0;
    IC=0;
    j=0;
   PI=0;
   SI=3;
    TI=0;
    // PT=new int[40];
    this.PTR=page*10;
    allocated=new boolean[30];
    alloc=new boolean[30];
}
private void startExecution() throws IOException {
    IC = 00;
    executeUserProgram();
}

public void executeUserProgram() throws IOException {
    IC=0;
    while(true){
```

```
int physicalAddress = addMap(IC);
IR = memory[physicalAddress];
IC++;



if(IR[0]=='L' && IR[1]=='R'){
    this.TLC++;
    String Line = new String(IR);
    int num=Integer.parseInt(Line.substring(2));
    if(allocated[num/10]==false)
    {
        PI=3;
        masterMode();
    }
    num=addMap(num);
    R[0]=memory[num][0];
    R[1]=memory[num][1];
    R[2]=memory[num][2];
    R[3]=memory[num][3];
}
else if(IR[0]=='S' && IR[1]=='R')
```

```
{
    this.TLC++;
    String Line = new String(IR);
    int num=Integer.parseInt(Line.substring(2));
    if(allocated[num/10]==false)
    {
        PI=3;
        masterMode();
    }
    num=addMap(num);
    memory[num][0]=R[0];
    memory[num][1]=R[1];
    memory[num][2]=R[2];
    memory[num][3]=R[3];
}
else if(IR[0]=='C' && IR[1]=='R')
{
    this.TLC++;
    String Line = new String(IR);

    int num=Integer.parseInt(Line.substring(2));
```

```
            num=addMap(num);
            if(memory[num][0]==R[0]&& memory[num][1]==R[1] &&
memory[num][2]==R[2]&& memory[num][3]==R[3])
            {
                T=1;
            }


        }
        else if(IR[0]=='B' && IR[1]=='T')
        {
            this.TLC++;
            if(T==1)
            {
                String LINE = new String(IR);
                int num=Integer.parseInt(LINE.substring(2));
               IC=num;
                T=0;
            }
        }
        else if(IR[0]=='G' && IR[1]=='D')
        {
```

```
            this.TLC+=2;
            this.SI = 1;
            masterMode();
        }
        else if(IR[0]=='P' && IR[1]=='D')
        {
            this.SI=2;
            masterMode();
            continue;
        }
        else if(IR[0]=='H' || IR[3]=='H')
        {
            this.SI=3;
            break;
        }



    }
}
```

```
public void masterMode(){
    //Case of TI and PI
    if(TI == 0 && PI == 1){terminate(4);}
    if(TI == 0 && PI == 2){terminate(5);}
    if(TI == 0 && PI==3){
        terminate(6);
    }
    if(TI == 2 && PI == 1){terminate(3);terminate(4);System.exit(0);}
    if(TI == 2 && PI == 2){terminate(3);terminate(5);System.exit(0);}
    if(TI == 2 && PI == 3){terminate(3);System.exit(0);}

    // Case of TI and SI
    if(TI == 0 && SI == 1){
        Read();
    }
    if(TI == 0 && SI == 2){
        try {
           Write();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
```

```
}
if(TI == 0 && SI == 3){terminate(0);System.exit(0);}
if(TI == 2 && SI == 1){terminate(3);System.exit(0);}
if(TI == 2 && SI == 2){
    try {
        Write();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    terminate(3);
    System.exit(0);
}
if(TI == 2 || SI == 3){
    if(TI==2){
        terminate(3);
        System.exit(0);
    }
    if(SI==3){
        terminate(0);
        System.exit(0);
    }
```

```
    }
}
public void Write() throws IOException {
    this.LLC++;
    if(this.LLC>this.TLL)
        terminate(2);
    String Line = new String(IR);
    int num=Integer.parseInt(Line.substring(2));
    String t = "",total="";
    if(!allocated[num / 10]){
        PI = 3;
        masterMode();

    }
    int realA = addMap(num);
    for(int i=0;i<10;i++)
    {

        t = new String(memory[realA+i]);

        t = t.trim();
```

```
        if (!t.isEmpty()) {
            total = total.concat(t);
        }
    }
    if( total.equals("")){
        PI = 3;
    }
    System.out.println(total);
    fw.write("\n"+total);
    fw.flush();

}

public void Read() {



    String Line = new String(IR);


    int num=Integer.parseInt(Line.substring(2));
    int realAddress=addMap(num);
    int temp=realAddress;
```

```
        try {
            Line=fread.readLine();


        } catch (IOException e) {
            e.printStackTrace();
        }
        buffer=Line.toCharArray();
        for (int i = 0; i < Line.length();) {


            memory[realAddress][(i%4)]=buffer[i];
            i++;
            if(i%4==0)
                realAddress++;
        }
        if (memory[temp][0] == '$' && memory[temp][1] == 'E' &&
memory[temp][2] == 'N' && memory[temp][3] == 'D') {
            terminate(1);
        }
    }
```

```
int allocate(){
    Random random = new Random();
    int page= random.nextInt(30);
    while(alloc[page])
        page =random.nextInt(30);

    alloc[page] = true;
    return page;
}
public void print_memory(){
    for(int i=0;i<300;i++) {
        System.out.println("memory["+i+"] "+new String(memory[i]));
    }
}
void terminate(int em){
    String str="\n\n";

    try{
        fw.write(str+"\n");
        fw.flush();
        switch (em) {
```

```
case 0:{
    str = ("No Error");
    break;
}
case 1:{
    str = ("Out of Data");
    break;
}
case 2:{
    str = ("Line Limit Exceeded");
    break;
}
case 3:{
    str = ("Time Limit Exceeded");
    break;
}
case 4:{
    str = ("Operation Code Error");
    break;
}
case 5:{
```

```
                str = ("Operand Error");
                break;
            }
            case 6:{
                str = ("Invalid Page Fault");
                break;
            }
            default:{
                str = ("No Exception Mentioned");
                break;
            }
        }
        fw.write(str+"\n");
        fw.flush();
        throw new MyException(str);
    }catch(MyException e){
        System.out.println("Error:- "+e.getMessage());
    }catch(IOException ie){
        System.out.println(ie);
    }
}
```

```
    public static void main(String[] arg)throws IOException{
        Main ph=new Main();
        ph.load();
    }
}
```

Input.text

```
$AMJ000100030015
GD10PD10H
$DTA
HELLO WORLD
$END0001
$AMJ000100130001
GD20GD30GD40GD50LR20CR30BT11PD50000HPD40H
$DTA
VIT
VIIT
SAME
NOT SAME
$END0001
$AMJ000100030001
GD20GD30GD40GD50PD20PD30LR20CR30BT11PD50000HPD40H
```

$DTA

Mona

Mona

SAME

NOT SAME

$END0001

$AMJ000100030003

GD20LR20SR45SR53SR57SR61SR65SR69PD40PD50PD60H

$DTA

*

$END0001

$AMJ000100030003

GD20LR20SR31SR41SR51SR52SR53PD30PD40PD50H

$DTA

:

$END0001

$AMJ000100030003

GD20GD30GD40PD20PD30PD40H

$DTA

HELLO

HOW ARE

YOU

$END0001

$AMJ000100030005

GD10GD20GD30GD40GD50PD10PD20PD30PD40PD50H

$DTA

5

4

3

2

1

$END0001

$AMJ000100030003

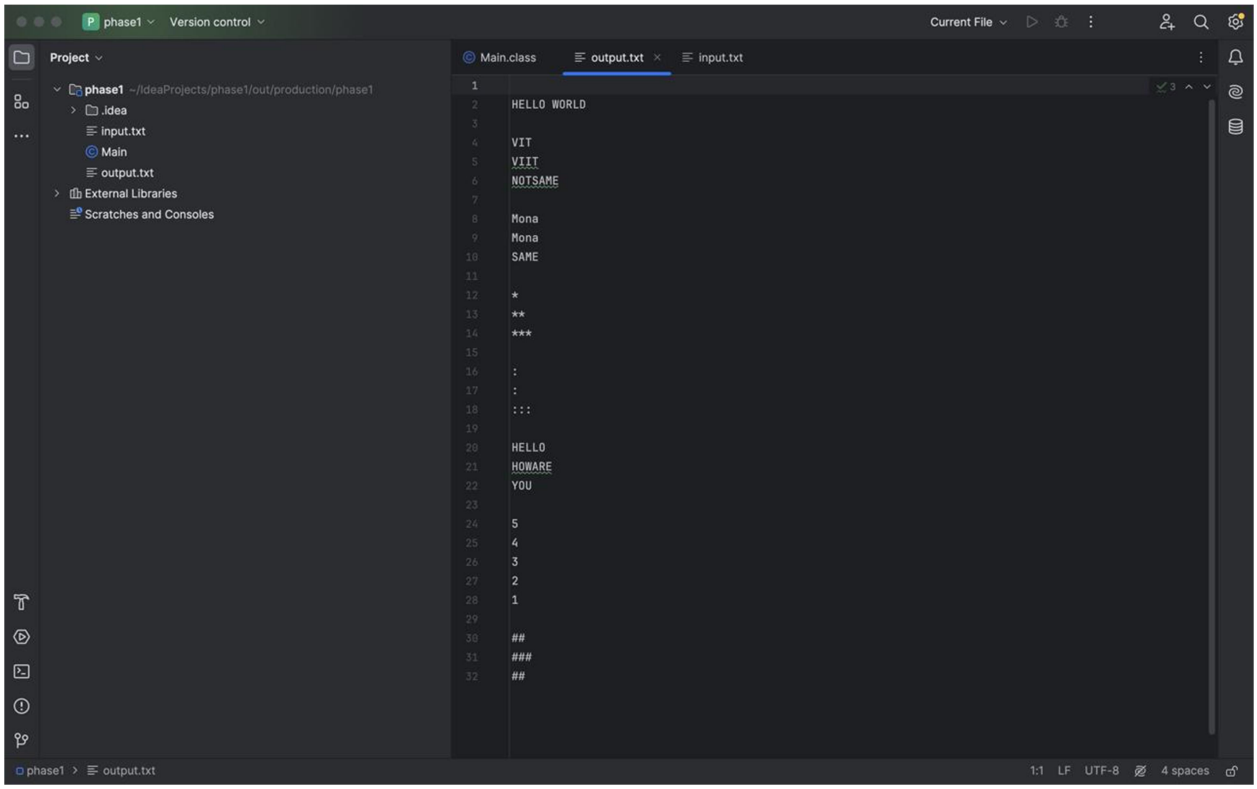
GD20LR20SR41SR43SR51SR52SR53SR61SR63PD40PD50PD60H

$DTA

#

$END0001

Output.txt


phase1
Version control
Current File
Project
phase1 ~/IdeaProjects/phase1/out/production/phase1
.idea
input.txt
Main
output.txt
External Libraries
Scratches and Consoles
Main.class
output.txt
input.txt
HELLO WORLD
VIT
VIIT
NOTSAME
Mona
Mona
SAME
*
**
***
:
:
:::
HELLO
HOWARE
YOU
5
4
3
2
1
##
###
##
phase1 > output.txt
1:1
LF
UTF-8
4 spaces