Assignment No: 1A

----------Obj.java----------

```
package firstjava;

public class obj {
	String name;
	int addr;
	obj(String nm, int address)
	{
		this.name=nm;
		this.addr=address;
	}
}
```

----------pooltable.java----------

```
package firstjava;

public class pooltable {
int first,total_literals;
public pooltable(int f, int l) {
	// TODO Auto-generated constructor stub
	this.first=f;
	this.total_literals=l;
}
}
```

----------spos.java----------

```
package firstjava;
import java.io.*;
import java.util.*;
class spos
{
	public static void main(String args[]) throws NullPointerException, FileNotFoundException
	{
	 String REG[] = {"ax","bx","cx","dx"};
	 String IS[]={"stop","add","sub","mult","mover","movem","comp","bc","div","read"};
	 String DL[]={"ds","dc"};
	 obj[] literal_table = new obj[10];
	 obj[] symb_table = new obj[10];
	 obj[] optab =new obj[60];
	 pooltable[] pooltab=new pooltable[5];
	 String line;
	 try{
		BufferedReader br=new BufferedReader(new FileReader("/home/student/eclipse-   workspace/firstjava/src/firstjava/sampal.txt"));
		 BufferedWriter bw=new BufferedWriter(new FileWriter("/home/student/eclipse-workspace/firstjava/src/firstjava/output.txt"));
		Boolean start=false;
		 Boolean end=false,fill_addr=false,ltorg=false;
	 int total_symb=0,total_ltr=0,optab_cnt=0,pooltab_cnt=0,loc=0,temp,pos;
	 while((line=br.readLine())!=null&&!end)
	    {
		 String tokens[]=line.split(" ",4);
		 if(loc!=0 && !ltorg)
		 {
			bw.write("\n"+String.valueOf(loc));
			ltorg=false;
			loc++;
		 }
		 ltorg=fill_addr=false;
		 for(int k=0;k<tokens.length;k++)
		 {
			pos = -1;
			if(start==true)
			{
				loc=Integer.parseInt(tokens[k]);
				start=false;
			}

			switch(tokens[k])
				{
				case "start" : start = true;
							pos = 1;
							bw.write("\t(AD,"+pos+")");
							break;
				case "end": end=true;
```

```
                                        pos = 2;
                                    bw.write("\t(AD,"+pos+")\n");
                                    for(temp=0;temp<total_ltr;temp++)
                                        if(literal_table[temp].addr==0)
                                        { literal_table[temp].addr=loc-1;
                                          bw.write("\t(DL,1)\t(C,"+literal_table[temp].name+")"+"\n"+loc++);
                                        }
                                   /* if(pooltab_cnt==0)
                                        pooltab[pooltab_cnt++]=new pooltable(0,temp);
                                    else
                                    {
pooltab[pooltab_cnt]=new pooltable(pooltab[pooltab_cnt-1].first+pooltab[pooltab_cnt-1].total_literals,total_ltr-pooltab[pooltab_cnt-1].first-1);
                                        pooltab_cnt++;
                                    } */
                                    break;

                        case "origin": pos = 3;
                                    bw.write("\t(AD,"+pos+")");
                                    pos= search(tokens[++k],symb_table,total_symb);
                                    k++;
                                    bw.write("\t(C,"+(symb_table[pos].addr)+")");
                                    loc = symb_table[pos].addr;
                                    break;

                         case "ltorg": ltorg=true;
                                                pos = 5;
                                    for(temp=0;temp<total_ltr;temp++)
                                        if(literal_table[temp].addr==0)
                                        {
                                            literal_table[temp].addr=loc-1;
                                    bw.write("\t(DL,1)\t(C,"+literal_table[temp].name+")"+"\n"+loc++);
                                        }
                                    if(pooltab_cnt==0)
                                        pooltab[pooltab_cnt++]=new pooltable(0,temp);
                                    else
                                    {
pooltab[pooltab_cnt]=new pooltable(pooltab[pooltab_cnt-1].first+pooltab[pooltab_cnt-1].total_literals,total_ltr-pooltab[pooltab_cnt-1].first-1);
                                        pooltab_cnt++;
                                    }
                                    break;

                         case "equ": pos = 4;
                                    bw.write("\t(AD,"+pos+")");
                                    String prev_token=tokens[k-1];
                                    int pos1 = search(prev_token,symb_table,total_symb);
                                    pos = search(tokens[++k],symb_table,total_symb);
                                    symb_table[pos1].addr = symb_table[pos].addr;
                                    bw.write("\t(S,"+(pos+1)+")");
                                    break;

                         }
                if(pos == -1)
                {
                         pos=search(tokens[k], IS);
                         if(pos != -1)
                         {
                         bw.write("\t(IS,"+(pos+1)+")");
                         optab[optab_cnt++]=new obj(tokens[k], pos);
                         }
                         else
                         {
                                  pos=search(tokens[k], DL);
                                  if(pos != -1)
                                  {
                                           bw.write("\t(DL,"+(pos+1)+")");
                                  optab[optab_cnt++]=new obj(tokens[k], pos);
                                           fill_addr=true;
                                  }
                                  else if(tokens[k].matches("[a-zA-Z]+:"))
                                  {
                                           pos = search(tokens[k], symb_table,total_symb);
```

```
                                                if(pos == -1)
                                                {
                                                        symb_table[total_symb++]=new obj(tokens[k].substring(0,tokens[k].length()-1),loc-1);
                                                        bw.write("\t(S,"+total_symb+")");
                                                        pos=total_symb;
                                                }
                                        }
                                }
                        }
                        if(pos == -1)
                        {
                                pos=search(tokens[k], REG);
                                if(pos!=-1)
                                        bw.write("\t(RG,"+(pos+1)+")");
                                else
                                {
                                        if(tokens[k].matches("=\\d+'"))
                                        {
                                                String s=tokens[k].substring(2, 3);

                                                literal_table[total_ltr++]=new obj(s, 0);
                                                bw.write("\t(L,"+total_ltr+")");
                                        }
                                        else if(tokens[k].matches("\\d+")||tokens[k].matches("\\d+H") || tokens[k].matches("\\d+h") ||
tokens[k].matches("\\d+D") || tokens[k].matches("\\d+d"))
                                                bw.write("\t(C,"+tokens[k]+")");
                                        else
                                        {
                                                pos = search(tokens[k], symb_table,total_symb);
                                                if(fill_addr && pos!=-1)
                                                        {
                                                                symb_table[pos].addr=loc-1;
                                                                fill_addr=false;
                                                        }
                                                else if(pos==-1)
                                                {
                                                        symb_table[total_symb++]=new obj(tokens[k],0);
                                                        bw.write("\t(S," + total_symb + ")");
                                                }
                                                else
                                                        bw.write("\t(S," + pos + ")");
                                        }
                                }
                        }
                }
        }
        System.out.println("\n*******************SYMBOL TABLE**********************");
        System.out.println("\nSYMBOL\tADDRESS");
        for(int i=0;i<total_symb;i++)
                System.out.println(symb_table[i].name+"\t"+symb_table[i].addr);

        pooltab[pooltab_cnt]=new pooltable(pooltab[pooltab_cnt-1].first+pooltab[pooltab_cnt-1].total_literals,total_ltr-pooltab[pooltab_cnt-1].first-2);
                pooltab_cnt++;

        System.out.println("\n*****************POOL TABLE*********************");
        System.out.println("\nPOOL\tTOTAL LITERALS");

        for(int i=0;i<pooltab_cnt;i++)
                System.out.println(pooltab[i].first+"\t"+pooltab[i].total_literals);

        System.out.println("\n**************LITERAL TABLE***************");
        System.out.println("\nIndex\tLITERAL\tADDRESS");
        for(int i=0;i<total_ltr;i++)
        {
                if(literal_table[i].addr==0)
                        literal_table[i].addr=loc++;
                System.out.println((i) +"\t"+literal_table[i].name+"\t"+literal_table[i].addr);
        }

        System.out.println("\n***************OPTABLE**********************");
        System.out.println("\nMNEMONIC\tOPCODE");
```

```
                for(int i=0;i<IS.length;i++)
                        System.out.println(IS[i]+"\t\t"+i);

                br.close();
                bw.close();
            }
            catch(Exception e)
            {
             System.out.println("error while reading the file");
             e.printStackTrace();
            }
            BufferedReader br=new BufferedReader(new FileReader("/home/student/eclipse-workspace/firstjava/src/firstjava/output.txt"));
            System.out.println("\n*********Output1.txt*****************\n");
            try {
                    while((line=br.readLine())!=null)
                            System.out.println(line);
                    br.close();
            } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }

            }
            public static int search(String token, String[] list) {
                    for(int i=0;i<list.length;i++)
                            if(token.equalsIgnoreCase(list[i]))
                                    return i;
                    return -1;
            }
            public static int search(String token, obj[] list,int cnt) {
                            for(int i=0;i<cnt;i++)
                                    if(token.equalsIgnoreCase(list[i].name))
                                            return i;
                    return -1;

            }
}
```

----------Input:----------

```
Samaple.txt
start 100
mover ax 05
mover bx 10
up: add ax bx
movem a ='5'
mult ax a
origin up
ltorg
movem b ='8'
movem c ='8'
ltorg
movem b ='7'
movem c ='8'
ds a 02
dc b 10
ds c 09
next equ up
end
```

Assignment NO: 1B

----------Obj.java----------

```
package secondjava;
public class obj {
	String name;
	int addr;
	obj(String nm, int address)
	{
		this.name=nm;
		this.addr=address;
	}
}
```

----------Pooltable.java----------

```
package secondjava;
public class pooltable {
int first,total_literals;
public pooltable(int f, int l) {
	// TODO Auto-generated constructor stub
	this.first=f;
	this.total_literals=l;
}
```

----------spos2.java----------

```
package secondjava;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class spos2 {
	static obj[] symb_table=new obj[10];
	static obj[] literal_table=new obj[10];
	static int symb_found=0;
	public static void main(String[] args) throws IOException {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);
		System.out.println("ENTER TOTAL NUMBER OF SYMBOLS : ");
		int total_symb = sc.nextInt();
		int pos,num;
		for(int i=0 ; i<total_symb; i++)
		{
			symb_table[i]=new obj("",0);
			System.out.println("ENTER SYMBOL NAME : ");
			symb_table[i].name=sc.next();
			System.out.println("ENTER SYMBOL ADDRESS : ");
			symb_table[i].addr=sc.nextInt();
		}
		System.out.println("ENTER TOTAL NUMBER OF LITRALS : ");
		int total_ltr = sc.nextInt();
		for(int i=0 ; i<total_ltr; i++)
		{
			literal_table[i]=new obj("",0);
			System.out.println("ENTER LITERAL NAME : ");
			literal_table[i].name=sc.next();
			System.out.println("ENTER LITERAL ADDRESS : ");
			literal_table[i].addr=sc.nextInt();
		}

		System.out.println("\n**********SYMBOL TABLE**********");
		System.out.println("\nSYMBOL\tADDRESS");
		for(int i=0;i<total_symb;i++)
			System.out.println(symb_table[i].name+"\t"+symb_table[i].addr);

		System.out.println("\n**********LITERAL TABLE**********");
	System.out.println("\nIndex\tLITERAL\tADDRESS");
	for(int i=0;i<total_ltr;i++)
		System.out.println((i+1) +"\t"+literal_table[i].name+"\t"+literal_table[i].addr);

		BufferedReader br2=new BufferedReader(new FileReader("/home/student/eclipse-workspace/secondjava/src/secondjava/output.txt"));
		String line;
		boolean symbol_error=false,undef_mnemonic=false;
		System.out.println("\n**********OUTPUT FILE**********\n\n");
```

```
lab: while((line = br2.readLine())!=null)
{
	String[] token_list=line.split("\\s+",5);
	symbol_error=undef_mnemonic=false;
	for(String token:token_list)
	{
		if(token.length()>0)
		{
			pos = -1;
			if(token.matches("---"))
			{
				System.out.print("\t---");
				undef_mnemonic=true;
			}
			else if(token.matches("[0-9]+"))
				System.out.print("\n\n"+token);
			else
			{
				String letters = token.replaceAll("[^A-Za-z]+", "");
				num = Integer.parseInt(token.replaceAll("[^0-9]+", ""));

				if(token.matches("\\([0-9]+\\)"))
					System.out.print("\t"+num);
				else
				{
					switch (letters.toUpperCase()) {
						case "S" : if(symb_table[num-1].addr==0)
							{
								System.out.print("\t---");
								symbol_error=true;
							}
							else
								System.out.print("\t"+symb_table[num-1].addr);
								break;
						case "L" : System.out.print("\t"+literal_table[num-1].addr);
								break;
						case "AD" : System.out.print("\n");
								continue lab;
						case "DL" :
							switch (num){
								case 1:	System.out.print("\n");
									break;
								case 2: System.out.print("\t 00 \t 00");
									} continue lab;
						case "C" : System.out.print("\t"+num);
								break;
						default: System.out.print("\t"+"00"+num);
					}
				}
			}
		}
	}
	if(symbol_error)
		System.out.print("\n\n*****************SYMBOL IS NOT DEFINED**************");
	if(undef_mnemonic)
		System.out.print("\n\n***************INVALID MNEMONIC *************");
}
int[] flag=new int[total_symb];
for(int i=0;i<total_symb;i++)
{
	symb_found=0;
	for(int j=0;j<total_symb;j++)
		if(symb_table[i].name.equalsIgnoreCase(symb_table[j].name) && flag[j]==0)
		{
			symb_found++;
			flag[i]=flag[j]=1;
		}
	if(symb_found>1)
		System.out.print("\n\n*************** '"+symb_table[i].name+"' IS DUPLICATE SYMBOL********");
}
}
```

}

----------input----------

Input1.txt	***Through console***
(AD,1) (C,100) 100 (IS,5) (RG,1) (C,05) 101 (IS,5) (RG,2) (C,10) 102 (S,1) (IS,2) (RG,1) (RG,2) 103 (IS,6) (S,2) (L,1) 104 (IS,4) (RG,1) (S,1) 105 (AD,3) (C,102) 102 (DL,1) (C,5) 103 (IS,6) (S,3) (L,2) 104 (IS,6) (S,4) (L,3) 105 (DL,1) (C,8) 106 (DL,1) (C,8) 107 (IS,6) (S,2) (L,4) 108 (IS,6) (S,3) (L,5) 109 (DL,1) (C,02) 110 (DL,2) (C,10) 111 (DL,1) (C,09) 112 (S,5) (AD,4) (S,1) 113 (AD,2) (DL,1) (C,7) 114 (DL,1) (C,8)	**No of literals**: 5 up 102 a 109 b 110 c 111 next 102 **No of Literals**: 5 5 102 8 105 8 106 7 113 8 114

Assignment NO: 2A

----------arglist.java----------

```
package thirdjava;

public class arglist {
	String argname;
	arglist(String argument) {
		// TODO Auto-generated constructor stub
		this.argname=argument;
	}
}
```

----------mdt.java----------

```
package thirdjava;

public class mdt {
String stmnt;
public mdt() {
	// TODO Auto-generated constructor stub
	stmnt="";
}
}
```

----------mnt.java----------

```
package thirdjava;

public class mnt {
	String name;
	int addr;
	int arg_cnt;
	mnt(String nm, int address)
	{
		this.name=nm;
		this.addr=address;
		this.arg_cnt=0;
	}
}
```

----------spos3.java----------

```
package thirdjava;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class spos3 {

	public static void main(String[] args) throws IOException {
		// TODO Auto-generated method stub
		BufferedReader br1=new BufferedReader(new FileReader("/home/student/eclipse-workspace/thirdjava/src/thirdjava/input3.txt"));
		String line;
		mdt[] MDT=new mdt[20];
		mnt[] MNT=new mnt[4];
		arglist[] ARGLIST = new arglist[10];
		boolean macro_start=false,macro_end=false,fill_arglist=false;
		int mdt_cnt=0,mnt_cnt=0,arglist_cnt=0;
		while((line = br1.readLine())!=null)
		{
			line=line.replaceAll(",", " ");
			String[] tokens=line.split("\\s+");
			MDT[mdt_cnt] = new mdt();
			String stmnt = "";
			for(int i=0;i<tokens.length;i++)
			{
				if(tokens[i].equalsIgnoreCase("mend"))
				{
					MDT[mdt_cnt++].stmnt = "\t"+tokens[i];
					macro_end = true;
				}
				if(tokens[i].equalsIgnoreCase("macro"))
				{
```

```
                        macro_start = true;
                        macro_end = false;
                }
                else if(!macro_end)
                {
                        if(macro_start)
                        {
                                MNT[mnt_cnt++]=new mnt(tokens[i],mdt_cnt);
                                macro_start=false;
                                fill_arglist=true;
                        }
                        if(fill_arglist)
                        {
                                while(i<tokens.length)
                                {
                                        MDT[mdt_cnt].stmnt = MDT[mdt_cnt].stmnt+ "\t" + tokens[i];
                                        stmnt = stmnt +"\t"+ tokens[i];
                                        if(tokens[i].matches("&[a-zA-Z]+")||tokens[i].matches("&[a-zA-Z]+[0-9]+"))
                                                ARGLIST[arglist_cnt++]=new arglist(tokens[i]);
                                        i++;
                                }
                                fill_arglist=false;
                        }
                        else
                        {

                                if(tokens[i].matches("[a-zA-Z]+") || tokens[i].matches("[a-zA-Z]+[0-9]+")||tokens[i].matches("[0-9]+"))
                                {
                                        MDT[mdt_cnt].stmnt = MDT[mdt_cnt].stmnt+ "\t" + tokens[i];
                                        stmnt = stmnt +"\t"+ tokens[i];
                                }
                                if(tokens[i].matches("&[a-zA-Z]+") || tokens[i].matches("&[a-zA-Z]+[0-9]+"))
                                {
                                        for(int j=0;j<arglist_cnt;j++)
                                                if(tokens[i].equals(ARGLIST[j].argname))
                                                {
                                                        MDT[mdt_cnt].stmnt = MDT[mdt_cnt].stmnt + "\t#"+(j+1);
                                                        stmnt = stmnt +"\t#"+(j+1);
                                                }
                                }
                        }
                }
        }
        if(stmnt!="" && !macro_end)
                mdt_cnt++;
}
br1.close();

BufferedWriter bw1=new BufferedWriter(new FileWriter("/home/student/eclipse-workspace/thirdjava/src/thirdjava/MNT.txt"));
System.out.println("\n\t********MACRO NAME TABLE**********");
System.out.println("\n\tINDEX\tNAME\tADDRESS");
for(int i=0;i<mnt_cnt;i++)
{
        System.out.println("\t"+i+"\t"+MNT[i].name+"\t"+MNT[i].addr);
        bw1.write(MNT[i].name+"\t"+MNT[i].addr+"\n");
}
bw1.close();

bw1=new BufferedWriter(new FileWriter("/home/student/eclipse-workspace/thirdjava/src/thirdjava/ARGLIST.txt"));
System.out.println("\n\n\t********ARGUMENT LIST**********");
System.out.println("\n\tINDEX\tNAME\tADDRESS");
for(int i=0;i<arglist_cnt;i++)
{
        System.out.println("\t"+i+"\t"+ARGLIST[i].argname);
        bw1.write(ARGLIST[i].argname+"\n");
}
bw1.close();

System.out.println("\n\t********MACRO DEFINITION TABLE**********");
System.out.println("\n\tINDEX\t\tSTATEMENT");
```

```
                bw1=new BufferedWriter(new FileWriter("/home/student/eclipse-workspace/thirdjava/src/thirdjava/MDT.txt"));
                for(int i=0;i<mdt_cnt;i++)
                {
                        System.out.println("\t"+i+"\t"+MDT[i].stmnt);
                        bw1.write(MDT[i].stmnt+"\n");
                }
                bw1.close();
        }
}
```

----------input----------

Input3.txt

```
MACRO
INCR &X,&Y,&REG1 = AREG
MOVER &REG1,&X
ADD &REG1,&Y
MOVEM &REG1,&X
MEND
MACRO
DECR &A,&B,&REG2 = BREG
MOVER &REG2,&A
SUB &REG2,&B
MOVEM &REG2,&A
MEND
START 100
READ N1
READ N2
DECR N1,N2
INCR N1,N2
STOP
N1 DS 1
N2 DS 2
END
```

----------Output----------

----------MDT.txt----------

```
INCR    &X      &Y      &REG1           =       AREG
        MOVER           #3      #1
        ADD     #3      #2
        MOVEM           #3      #1
        MEND
        DECR    &A      &B      &REG2           =
        BREG
        MOVER           #6      #4
        SUB     #6      #5
        MOVEM           #6      #4
        MEND
```

----------Output----------

----------MNT.txt----------

INCR 0

DECR 5

----------ARG.txt----------

```
&X
&Y
&REG1
&A
&B
&REG2
```

Assignment NO: 2-B

----------arglist.java----------

```
package fourthjava;

public class arglist {
	String argname;
	arglist(String argument) {
		// TODO Auto-generated constructor stub
		this.argname=argument;
	}
}
```

----------mdt.java----------

```
package fourthjava;

public class mdt {
String stmnt;
public mdt() {
	// TODO Auto-generated constructor stub
	stmnt="";
}
}
```

----------mnt.java----------

```
package fourthjava;

public class mnt {
	String name;
	int addr;
	int arg_cnt;
	mnt(String nm, int address)
	{
		this.name=nm;
		this.addr=address;
		this.arg_cnt=0;
	}
}
```

----------spos4.java----------

```
package fourthjava;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class spos4 {

	public static void main(String[] args) throws IOException {
		// TODO Auto-generated method stub
		mdt[] MDT=new mdt[20];
		mnt[] MNT=new mnt[4];
		arglist[] formal_parameter=new arglist[10];
		int macro_addr = -1;

		boolean macro_start=false,macro_end=false;
		int macro_call = -1;
		int mdt_cnt=0,mnt_cnt=0,formal_arglist_cnt=0,actual_arglist_cnt=0,temp_cnt=0,temp_cnt1=0;

		BufferedReader br1=new BufferedReader(new FileReader("/home/student/eclipse-workspace/fourthjava/src/MNT.txt"));
		String line;
		while((line = br1.readLine())!=null)
		{
			String[] parts=line.split("\\s+");
			MNT[mnt_cnt++]=new mnt(parts[0], Integer.parseInt(parts[1]),Integer.parseInt(parts[2]));
		}
		br1.close();
		System.out.println("\n\t********MACRO NAME TABLE**********");
		System.out.println("\n\tINDEX\tNAME\tADDRESS\tTOTAL ARGUMENTS");
		for(int i=0;i<mnt_cnt;i++)
			System.out.println("\t"+i+"\t"+MNT[i].name+"\t"+MNT[i].addr+"\t\t"+MNT[i].arg_cnt);
```

```
br1=new BufferedReader(new FileReader("/home/student/eclipse-workspace/fourthjava/src/fourthjava/ARG.txt"));
while((line = br1.readLine())!=null)
{
		String[] parameters=line.split("\\s+");
		formal_parameter[formal_arglist_cnt++]=new arglist(parameters[0]);
		if(parameters.length>1)
			formal_parameter[formal_arglist_cnt-1].value = parameters[1];
}
br1.close();

System.out.println("\n\n\t********FORMAL ARGUMENT LIST**********");
System.out.println("\n\tINDEX\tNAME\tADDRESS");
for(int i=0;i<formal_arglist_cnt;i++)
		System.out.println("\t"+i+"\t"+formal_parameter[i].argname+"\t"+formal_parameter[i].value);

br1=new BufferedReader(new FileReader("/home/student/eclipse-workspace/fourthjava/src/MDT.txt"));
while((line = br1.readLine())!=null)
{
		MDT[mdt_cnt]=new mdt();
		MDT[mdt_cnt++].stmnt=line;
}
br1.close();

System.out.println("\n\t********MACRO DEFINITION TABLE**********");
System.out.println("\n\tINDEX\t\tSTATEMENT");
for(int i=0;i<mdt_cnt;i++)
		System.out.println("\t"+i+"\t"+MDT[i].stmnt);

br1=new BufferedReader(new FileReader("/home/student/eclipse-workspace/fourthjava/src/fourthjava/input.txt"));
arglist[] actual_parameter=new arglist[10];

BufferedWriter bw1 = new BufferedWriter(new FileWriter("/home/student/eclipse-workspace/fourthjava/src/output.txt"));
while((line = br1.readLine())!=null)
{
		line=line.replaceAll(",", " ");
		String[] tokens=line.split("\\s+");
		temp_cnt1=0;
		for(String current_token:tokens)
		{
			if(current_token.equalsIgnoreCase("macro"))
			{
				macro_start=true;
				macro_end=false;
			}
			if(macro_end && !macro_start)
			{
				if(macro_call != -1 && temp_cnt<formal_arglist_cnt-1)
				{
				   if(formal_parameter[actual_arglist_cnt].value != "")
					actual_parameter[actual_arglist_cnt++]=new arglist(formal_parameter[actual_arglist_cnt-1].value);

					actual_parameter[actual_arglist_cnt++]=new arglist(current_token);

					if(formal_parameter[actual_arglist_cnt].value != "")
					actual_parameter[actual_arglist_cnt++]=new arglist(formal_parameter[actual_arglist_cnt-1].value);

				}
				for(int i=0;i<mnt_cnt;i++)
				{
					if(current_token.equals(MNT[i].name))
					{
						macro_call=i;
						temp_cnt1 = temp_cnt1 +MNT[i].arg_cnt;
						break;
					}
					temp_cnt1 = temp_cnt1 + MNT[i].arg_cnt;
				}
				if(macro_call == -1)
					bw1.write("\t" + current_token);
			}
			if(current_token.equalsIgnoreCase("mend"))
```

```
                                        {
                                                macro_end=true;
                                                macro_start=false;
                                        }
                                }
                                if(macro_call != -1)
                                {
                                        macro_addr=MNT[macro_call].addr+1;
                                        while(true)
                                        {
                                                if(MDT[macro_addr].stmnt.contains("mend") || MDT[macro_addr].stmnt.contains("MEND"))
                                                {
                                                        macro_call = -1;
                                                        break;
                                                }
                                                else
                                                {
                                                        bw1.write("\n");
                                                        String[] temp_tokens=MDT[macro_addr++].stmnt.split("\\s+");
                                                        for(String temp:temp_tokens)
                                                        {
                                                                if(temp.matches("#[0-9]+"))
                                                                {
                                                                        int num = Integer.parseInt(temp.replaceAll("[^0-9]+", ""));
                                                                        bw1.write(actual_parameter[num-1].argname+"\t");
                                                                }
                                                                else
                                                                        bw1.write(temp + "\t");
                                                        }
                                                }
                                        }
                                }
                                if(!macro_start )
                                        bw1.write("\n");
                                macro_call= -1;
                        }
                        br1.close();
                        bw1.close();

                        System.out.println("\n\n\t********ACTUAL ARGUMENT LIST**********");
                        System.out.println("\n\tINDEX\tNAME\tADDRESS");
                        for(int i=0;i<actual_arglist_cnt;i++)
                                System.out.println("\t"+i+"\t"+actual_parameter[i].argname);
                }
}
```

---------- input ----------

```
MACRO
INCR &X,&Y,&REG1
MOVER &REG1,&X
ADD &REG1,&Y
MOVEM &REG1,&X
MEND
MACRO
DECR &A,&B,&REG2
MOVER &REG2,&A
SUB &REG2,&B
MOVEM &REG2,&A
MEND
START 100
READ N1
READ N2
INCR N1,N2
DECR N1,N3
STOP
N1 DS 1
N2 DS 2
N3 DS 1
END
```

---------- Output.txt ----------

```
START   100
        READ    N1
        READ    N2

        MOVER   AREG    N1
        ADD     AREG    N2
        MOVEM   AREG    N1

        MOVER   BREG    N1
        SUB     BREG    N3
        MOVEM   BREG    N1
        STOP
N1      DS      1
N2      DS      2
N3      DS      1
        END
```

Assignment No: 3

----------b1.c----------

```
import java.io.*;
import java.util.*;
class B1 {

    static {
       System.loadLibrary("B1");
    }

    private native int add(int a, int b);
    private native int sub(int a, int b);
    private native int mult(int a, int b);
    private native int div(int a, int b);

    public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            int a, b,ch;
            System.out.println("\nEnter value of a : ");
            a = sc.nextInt();
            System.out.println("\nEnter value of b : ");
            b = sc.nextInt();
            do
            {
                    System.out.println("\nENTER YOUR CHOICE : ");
                    ch = sc.nextInt();

                    switch(ch)
                    {
                            case 1 : new B1().add(a,b);
                                    break;

                            case 2 : new B1().sub(a,b);
                                    break;

                            case 3 : new B1().mult(a,b);
                                    break;

                            case 4 : new B1().div(a,b);
                                    break;

                            default : System.out.println("Your choice is wrong.");
                    }
            }while(ch<5);
    }
}
```

----------B1.java----------

```
#include <jni.h>
 #include <stdio.h>
 #include "B1.h"

 JNIEXPORT int JNICALL Java_B1_add(JNIEnv *env, jobject obj, jint a, jint b)
 {
            printf("\n%d + %d = %d\n",a,b,(a+b));
            return;
 }

 JNIEXPORT int JNICALL Java_B1_sub(JNIEnv *env, jobject obj, jint a, jint b)
 {
    printf("\n%d - %d = %d\n",a,b,(a-b));
    return;
 }

 JNIEXPORT int JNICALL Java_B1_mult(JNIEnv *env, jobject obj, jint a, jint b)
 {
    printf("\n%d * %d = %d\n",a,b,(a*b));
    return;
```

```
}

JNIEXPORT int JNICALL Java_B1_div(JNIEnv *env, jobject obj, jint a, jint b)
{
   printf("\n%d / %d = %d\n",a,b,(a/b));
   return;
}
```

Exeution Steps:

$ **javac B1.java**
javah -classpath . B1
$ **ls**
B1.c B1.c~ B1.class B1.h B1.java
$ **gcc -shared -fPIC -I/usr/lib/jvm/default-java/include -I/usr/lib/jvm/default-java/include/linux B1.c -o libB1.so**
$ ls
B1.c B1.c~ B1.class B1.h B1.java libB1.so
$ **java -classpath . -Djava.library.path=. B1**
Hello World!

Assignment No. 5

----------FCFS.java----------

```
import java.io.*;

public class FCFS {

	public static void main(String args[]) throws Exception
		{
		int n,at[],bt[],wt[],tat[],ft[];
		float awt=0,att=0;
		InputStreamReader isr=new InputStreamReader(System.in);
		BufferedReader br=new BufferedReader(isr);
		System.out.println("Enter no of process");
		n=Integer.parseInt(br.readLine());
		ft=new int[n];
		wt=new int[n];
		tat=new int[n];
		bt=new int[n];
		at=new int[n];
		System.out.println("Enter Burst time for each process\n");
		for(int i=0;i<n;i++)
		{
		System.out.println("Process["+(i+1)+"]");
		bt[i]=Integer.parseInt(br.readLine());
		}
		System.out.println("\n\nEnter Arrival Time");
		for(int i=0;i<n;i++)
		{
		System.out.println("Process["+i+"]");
		at[i]=Integer.parseInt(br.readLine());
		}
		System.out.println("\n");
		wt[0]=0;
		ft[0]=bt[0]-at[0];
		for(int i=1;i<n;i++)
		{
		ft[i]=ft[i-1]+bt[i];
		wt[i]=ft[i-1];
		wt[i]=wt[i]-at[i];
		awt=awt+wt[i];
		}
		for(int i=0;i<n;i++)
		{
		tat[i]=wt[i]+bt[i];
		att=att+tat[i];
		}
		System.out.println("PROCESS\t\tBURST-TIME\tWAITING-TIME\tTURN AROUND-TIME\n");
		for(int i=0;i<n;i++)
		{
		System.out.println(" "+ i + "\t\t"+bt[i]+"\t\t\t"+wt[i]+"\t\t"+tat[i]);
		}
		awt=awt/n;
		att=att/n;
		System.out.println("\n");
		System.out.println("Avg waiting time="+awt+"\n");
		System.out.println("\n");
		System.out.println("Avg Turn around time="+att+"\n");
		}


	}
```

----------input----------

----------SJF.java----------

```
import java.util.Scanner;
// --------------------Non Preemptive SJF-------------------------

class SJF
{
public static void main(String args[]){
int burst_time[],process[],waiting_time[],arrival_time[],tat[];
int ft[],i,j,n,total=0,pos,temp;
float wait_avg,TAT_avg;
@SuppressWarnings("resource")
Scanner s = new Scanner(System.in);

System.out.print("Enter number of process: ");
n = s.nextInt();
ft=new int[n];
process = new int[n];
burst_time = new int[n];
waiting_time = new int[n];
arrival_time=new int[n];
tat = new int[n];

System.out.println("\nEnter Burst time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
burst_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}
System.out.println("\nEnter Arrival time:");
for(i=0;i<n;i++)
{
System.out.print("\nProcess["+(i+1)+"]: ");
arrival_time[i] = s.nextInt();;
process[i]=i+1; //Process Number
}

//Sorting

for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(arrival_time[j]<arrival_time[pos])
if(burst_time[j]<burst_time[pos])
pos=j;
}

temp=burst_time[i];
burst_time[i]=burst_time[pos];
burst_time[pos]=temp;

temp=process[i];
process[i]=process[pos];
process[pos]=temp;
}


//First process has 0 waiting time
waiting_time[0]=0;
ft[0]=burst_time[0]-arrival_time[0];
//calculate waiting time
for(i=1;i<n;i++)
{
```

```
waiting_time[i]=0;
for(j=0;j<i;j++)
{
ft[i]=ft[i-1]-arrival_time[i];
waiting_time[i]+=burst_time[j];
}
total+=waiting_time[i];
}


//Calculating Average waiting time
wait_avg=(float)total/n;
total=0;

System.out.println("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
tat[i]=burst_time[i]+waiting_time[i]; //Calculating Turnaround Time
total+=tat[i];
System.out.println("\n p"+process[i]+"\t\t "+burst_time[i]+"\t\t "+waiting_time[i]+"\t\t "+tat[i]);
}

//Calculation of Average Turnaround Time
TAT_avg=(float)total/n;
System.out.println("\n\nAverage Waiting Time: "+wait_avg);
System.out.println("\nAverage Turnaround Time: "+TAT_avg);

}
}
```

----------Priority.java----------

```
import java.util.Scanner;

public class Priority {

   public static void main(String args[]) {
        Scanner s = new Scanner(System.in);

        int x,n,p[],pp[],bt[],w[],t[],ft[],awt,atat,i;
          System.out.print("Enter the number of process : ");
           n = s.nextInt();
        p = new int[n];
        pp = new int[n];
        bt = new int[n];
        w = new int[n];
        t = new int[n];
             ft=new int[n];
  //n is number of process
  //p is process
  //pp is process priority
  //bt is process burst time
  //w is wait time
  // t is turnaround time
  //awt is average waiting time
  //atat is average turnaround time



   System.out.print("\n\t Enter burst time : time priorities \n");

  for(i=0;i<n;i++)
```

```
    {
       System.out.print("\nProcess["+(i+1)+"]:");
      bt[i] = s.nextInt();
      pp[i] = s.nextInt();
      p[i]=i+1;
    }

//sorting on the basis of priority
  for(i=0;i<n-1;i++)
   {
     for(int j=i+1;j<n;j++)
     {
       if(pp[i]<pp[j])
       {
     x=pp[i];
     pp[i]=pp[j];
     pp[j]=x;
     x=bt[i];
     bt[i]=bt[j];
     bt[j]=x;
     x=p[i];
     p[i]=p[j];
     p[j]=x;
      }
   }
}
w[0]=0;
awt=0;
t[0]=bt[0];
ft[0]=bt[0];
atat=t[0];
for(i=1;i<n;i++)
 {
ft[i]=ft[i-1]+bt[i];
w[i]=ft[i-1];
  // w[i]=t[i-1];
   awt+=w[i];
   t[i]=w[i]+bt[i];
   atat+=t[i];
 }

//Displaying the process

  System.out.print("\n\nProcess \t Burst Time \t Wait Time \t Turn Around Time   Priority \n");
for(i=0;i<n;i++)
  System.out.print("\n   "+p[i]+"\t\t   "+bt[i]+"\t\t     "+w[i]+"\t\t     "+t[i]+"\t\t     "+pp[i]+"\n");
awt/=n;
atat/=n;
  System.out.print("\n Average Wait Time : "+awt);
  System.out.print("\n Average Turn Around Time : "+atat);

       }
}
```

Assignment NO. 8

----------Bankers.java----------

```
import java.util.Scanner;
public class Bankers
{
   private int need[][],allocate[][],max[][],avail[][],np,nr;

   private void input()
   {
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter no. of processes and resources : ");
    np=sc.nextInt();  //no. of process
    nr=sc.nextInt();  //no. of resources
    need=new int[np][nr];  //initializing arrays
    max=new int[np][nr];
    allocate=new int[np][nr];
    avail=new int[1][nr];

    System.out.println("Enter allocation matrix -->");
    for(int i=0;i<np;i++)
         for(int j=0;j<nr;j++)
        allocate[i][j]=sc.nextInt();  //allocation matrix

    System.out.println("Enter max matrix -->");
    for(int i=0;i<np;i++)
         for(int j=0;j<nr;j++)
        max[i][j]=sc.nextInt();  //max matrix

       System.out.println("Enter available matrix -->");
       for(int j=0;j<nr;j++)
        avail[0][j]=sc.nextInt();  //available matrix

       sc.close();
   }

   private int[][] calc_need(){
     for(int i=0;i<np;i++)
       for(int j=0;j<nr;j++)  //calculating need matrix
        need[i][j]=max[i][j]-allocate[i][j];

     return need;
   }

   private boolean check(int i){
     //checking if all resources for ith process can be allocated
     for(int j=0;j<nr;j++)
     if(avail[0][j]<need[i][j])
        return false;

   return true;
   }

   public void isSafe()
   {
     input();
     calc_need();
     boolean done[]=new boolean[np];
     int j=0;

     while(j<np){  //until all process allocated
     boolean allocated=false;
     for(int i=0;i<np;i++)
      if(!done[i] && check(i)){  //trying to allocate
          for(int k=0;k<nr;k++)
```

```
            avail[0][k]=avail[0][k]-need[i][k]+max[i][k];
          System.out.println("Allocated process : "+i);
          allocated=done[i]=true;
               j++;
             }
           if(!allocated) break;  //if no allocation
        }
        if(j==np)  //if all processes are allocated
         System.out.println("\nSafely allocated");
        else
         System.out.println("All proceess cant be allocated safely");
    }

    public static void main(String[] args)
    {
      new Bankers().isSafe();
    }
}
/* ----------------output--------------------
Enter no. of processes and resources :
3
4
Enter allocation matrix -->
1 2 2 1
1 0 3 3
1 2 1 0
Enter max matrix -->
3 3 2 2
1 1 3 4
1 3 5 0
Enter available matrix -->
3 1 1 2
Allocated process : 0
Allocated process : 1
Allocated process : 2

Safely allocated
* /
```

Assignment No. 7

----------Optimal.java----------

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class Optimal
{
public static void main(String[] args) throws IOException, InterruptedException
{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
int frames, pointer = 0, hit = 0, fault = 0,ref_len;
boolean isFull = false;
int buffer[];
int reference[];
int mem_layout[][];
System.out.println("Please enter the number of Frames: ");
frames = Integer.parseInt(br.readLine());
System.out.println("Please enter the length of the Reference string:");
ref_len = Integer.parseInt(br.readLine());
reference = new int[ref_len];
mem_layout = new int[ref_len][frames];
buffer = new int[frames];
for(int j = 0; j < frames; j++)
buffer[j] = -1;
System.out.println("Please enter the reference string: ");
for(int i = 0; i < ref_len; i++)
{
reference[i] = Integer.parseInt(br.readLine());
}
System.out.println();
for(int i = 0; i < ref_len; i++)
{
	int search = -1;
	for(int j = 0; j < frames; j++)
	{
		if(buffer[j] == reference[i])
		{
			search = j;
			hit++;
			break;
		}
	}

	if(search == -1)
	{
		if(isFull)
		{
			int index[] = new int[frames];
			boolean index_flag[] = new boolean[frames];
			for(int j = i + 1; j < ref_len; j++)
			{
				for(int k = 0; k < frames; k++)
				{
					if((reference[j] == buffer[k]) && (index_flag[k] == false))
					{
						index[k] = j;
						index_flag[k] = true;
						break;
					}
				}
			}
			int max = index[0];
			pointer = 0;
			if(max == 0)
```

```
                                max = 200;
                        for(int j = 0; j < frames; j++)
                        {
                                if(index[j] == 0)
                                        index[j] = 200;
                                if(index[j] > max)
                                {
                                        max = index[j];
                                        pointer = j;
                                }
                        }
                }
                buffer[pointer] = reference[i];
                fault++;
                if(!isFull)
                {
                        pointer++;
                        if(pointer == frames)
                        {
                                pointer = 0;
                                isFull = true;
                        }
                }
        }

        for(int j = 0; j < frames; j++)
                mem_layout[i][j] = buffer[j];
}

System.out.println("1st frame    2nd frame     3rd frame");
for(int j = 0; j < ref_len; j++)
{
        for(int i = 0; i < frames; i++)
        {
                Thread.sleep(1500);
                System.out.printf("%3d \t\t",mem_layout[j][i]);
        }
        System.out.println();
}
        System.out.println("The number of Hits: " + hit);
        System.out.println("Hit Ratio: " + (float)((float)hit/ref_len));
        System.out.println("The number of Faults: " + fault);
        }
}
/*----------------OUTPUT------------------------------
Please enter the number of Frames:
3
Please enter the length of the Reference string:
6
Please enter the reference string:
1       2       3       4       1       5

1st frame    2nd frame     3rd frame
  1                -1               -1
  1                 2               -1
  1                 2                3
  1                 4                3
  1                 4                3
  5                 4                3
The number of Hits: 1
Hit Ratio: 0.16666667
The number of Faults: 5
*/
```

----------fifo.java----------

```
import java.io.*;
public class fifo {
public static void main(String[] args) throws IOException
{
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
int frames, pointer = 0, hit = 0, fault = 0,ref_len;
int buffer[];
int reference[];
int mem_ayout[][];
System.out.println("Please enter the number of Frames: ");
frames = Integer.parseInt(br.readLine());
System.out.println("Please enter the length of the Reference string: ");
ref_len = Integer.parseInt(br.readLine());
reference = new int[ref_len];
mem_layout = new int[ref_len][frames];
buffer = new int[frames];
for(int j = 0; j < frames; j++)
buffer[j] = -1;
System.out.println("Please enter the reference string: ");
for(int i = 0; i < ref_len; i++)
{
reference[i] = Integer.parseInt(br.readLine());
}
System.out.println();
for(int i = 0; i < ref_len; i++)
{
		int search = -1;
		for(int j = 0; j < frames; j++)
		{
			if(buffer[j] == reference[i])
			{
				search = j;
				hit++;
				break;
			}
		}

if(search == -1)
		{
			buffer[pointer] = reference[i];
			fault++;
			pointer++;
			if(pointer == frames)
			pointer = 0;
		}

		for(int j = 0; j < frames; j++)
		mem_layout[i][j] = buffer[j];
}

		for(int i = 0; i < frames; i++)
		{
		   for(int i = 0; i < frames; i++)
		   {
			Thread.sleep(1500);
			System.out.printf("%3d \t\t",mem_layout[j][i]);
		   }
		  System.out.println();
		}

System.out.println("The number of Hits: " + hit);
System.out.println("Hit Ratio: " + (float)((float)hit/ref_len));
```

```
System.out.println("The number of Faults: " + fault);
}
}

/*----------------OUTPUT------------------------------
Please enter the number of Frames:
3
Please enter the length of the Reference string:
6
Please enter the reference string:
1          2          3          4          1          5

1st frame    2nd frame     3rd frame
  1                  -1                 -1
  1                   2                 -1
  1                   2                  3
  4                   2                  3
  4                   1                  3
  4                   1                  5
The number of Hits: 0
Hit Ratio: 0.0
The number of Faults: 6
*/
```

Assignment No. 4

----------Reader Writer Problem----------

```
import java.util.concurrent.Semaphore;

class ReaderWritersProblem {

   static Semaphore readLock = new Semaphore(1);
   static Semaphore writeLock = new Semaphore(1);
   static int readCount = 0;

   static class Read implements Runnable {
      @Override
      public void run() {
         try {
            //Acquire Section
            readLock.acquire();
            readCount++;
            if (readCount == 1) {
               writeLock.acquire();
            }
            readLock.release();

            //Reading section
            System.out.println("Thread "+Thread.currentThread().getName() + " is READING");
            Thread.sleep(1500);
            System.out.println("Thread "+Thread.currentThread().getName() + " has FINISHED READING");

            //Releasing section
            readLock.acquire();
            readCount--;
            if(readCount == 0) {
               writeLock.release();
            }
            readLock.release();
         } catch (InterruptedException e) {
            System.out.println(e.getMessage());
         }
      }
   }

   static class Write implements Runnable {
      @Override
      public void run() {
         try {
            writeLock.acquire();
            System.out.println("Thread "+Thread.currentThread().getName() + " is WRITING");
            Thread.sleep(2500);
            System.out.println("Thread "+Thread.currentThread().getName() + " has finished WRITING");
            writeLock.release();
         } catch (InterruptedException e) {
            System.out.println(e.getMessage());
         }
      }
   }

   public static void main(String[] args) throws Exception {
      Read read = new Read();
      Write write = new Write();
      Thread t1 = new Thread(read);
      t1.setName("thread1");
      Thread t2 = new Thread(read);
      t2.setName("thread2");
      Thread t3 = new Thread(write);
      t3.setName("thread3");
      Thread t4 = new Thread(read);
```

```
        t4.setName("thread4");
        t1.start();
        t3.start();
        t2.start();
        t4.start();
    }
}
```

Assignment No: 7

```
---------------------------------------- Memory Placement ----------------------------------------------
import java.io.*;
import java.util.*;

public class MemoryAllocationAlgo {

        static int job[];
        static int block[];
        static int js,bs;
        static Scanner input=new Scanner(System.in);
        static int Allocation[];
        public static void main(String args[])
        {
                MemoryAllocationAlgo MA=new MemoryAllocationAlgo();
                while(true)
                {
                System.out.println("Menu:");
                System.out.println("\n1.Read Data-Job No. & Size, Block No. & Size \n2.First Fit \n3.Best Fit \n4.WorstFit\n5.Exit");
           System.out.println("Enter Your Choice:");
                int ch=Integer.parseInt(input.nextLine());
                switch(ch)
                {
                        case 1: System.out.println("\n Enter total no. of jobs:");
                                        js=Integer.parseInt(input.nextLine());
                                        System.out.println("\n Enter total no. of blocks:");
                                        bs=Integer.parseInt(input.nextLine());
                                        job=new int[js];
                                        block=new int[bs];
                                 MA.ReadData(js,bs);
                                break;
                        case 2:
                                MA.FirstFit();
                                break;
                        case 3:MA.BestFit();
                                break;
                        case 4:
                                break;
                        case 5:System.exit(0);
                                break;
                }//end of swith
            }//enf of while

 }//end of main
 void ReadData(int n,int m)
 {
                for(int i=0;i<n;i++)
                {
                  System.out.println("Enter Job Size:");
                           job[i]=Integer.parseInt(input.nextLine());
                }
                        for(int i=0;i<m;i++)
                {
                                System.out.println("Enter Block Size:");
                                 block[i]=Integer.parseInt(input.nextLine());
                }

 }
 void FirstFit()
 {   int flag=0;
             Allocation=new int[js];
             for (int i = 0; i < Allocation.length; i++)
        Allocation[i] = -1;
             for (int i = 0; i < js; i++)
    {
```

```
        for (int j = 0; j < bs; j++)
        {   flag=0;
            if (block[j] >=job[i])
            {  System.out.println("i="+i+" j="+j+" B="+block[j]+" J="+job[i]+" all="+Allocation[i]);
               for(int k=0;k<js;k++)
               {
                        if(Allocation[k]==j)
                                flag=1;
               }
               // allocate block j to p[i] process
               if(flag==0)
               {  Allocation[i] = j;
                  System.out.println(j+" B="+block[j]+" J="+job[i]+" all="+Allocation[i]);
                  break;
               }
            }
        }
    }
            Display();
}

void Display()
{
            System.out.println("\tJob No.   \tJobSize  \tBlock No.   \tFragment");
            for(int i=0;i<js;i++)
            {
                        System.out.print(" "+i+"\t "+job[i]+"\t ");
               if(Allocation[i]!=-1)
               {
                        System.out.print("\t"+Allocation[i]+"\t"+(block[Allocation[i]]-job[i]));
               }
               else
               {
                        System.out.println(" Not allocated");
               }
               System.out.println();

            }
}

void BestFit()
{
            int flag=0;
            Allocation=new int[js];
            for (int i = 0; i < Allocation.length; i++)
        Allocation[i] = -1;
            for (int i = 0; i < js; i++)
            {
                     int BestInd=-1;
                  for (int j = 0; j < bs; j++)
                  {
                     flag=0;
                    if (block[j] >=job[i])
                    {
                                for(int k=0;k<js; k++)
                                 {
                                        if(Allocation[k]==j)
                                        { flag=1; break;}
                                   }

                      if(BestInd==-1 && flag==0)
                        {
                                BestInd=j;
                         }
                    else if(flag==0 && block[BestInd]>block[j])
```

```
                    {
                                BestInd=j;
                    }
                    else
                    {
                      continue;
                    }
                }
        }
                    if(BestInd!=-1)
                {
                     Allocation[i]=BestInd;
                }
    }
        Display();
}


        void WorstFit()
        {

        }

}
```