```
/*
Problem Statement: Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented
features. The output of assignment-1 (intermediate file and symbol table) should be
input for this assignment.
*/
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;

public class Pass2 {
	public static void main(String[] Args) throws IOException{
		BufferedReader b1 = new BufferedReader(new FileReader("intermediate.txt"));
	    BufferedReader b2 = new BufferedReader(new FileReader("symtab.txt"));
	    BufferedReader b3 = new BufferedReader(new FileReader("littab.txt"));
	    FileWriter f1 = new FileWriter("Pass2.txt");
	    HashMap<Integer, String> symSymbol = new HashMap<Integer, String>();
	    HashMap<Integer, String> litSymbol = new HashMap<Integer, String>();
	    HashMap<Integer, String> litAddr = new HashMap<Integer, String>();
	    String s;
	    int symtabPointer=1,littabPointer=1,offset;
	    while((s=b2.readLine())!=null){
	        String word[]=s.split("\t\t\t");
	        symSymbol.put(symtabPointer++,word[1]);
	    }
	    while((s=b3.readLine())!=null){
	        String word[]=s.split("\t\t");
	        litSymbol.put(littabPointer,word[0]);
	        litAddr.put(littabPointer++,word[1]);
	    }
	    while((s=b1.readLine())!=null){
	        if(s.substring(1,6).compareToIgnoreCase("IS,00")==0){
	            f1.write("+00 0 000\n");
	        }
	        else if(s.substring(1,3).compareToIgnoreCase("IS")==0){
	            f1.write("+"+s.substring(4,6)+" ");
	            if(s.charAt(9)==')'){
	                f1.write(s.charAt(8)+" ");
	                offset=3;
	            }
```

```
                        else{
                                f1.write("0 ");
                                offset=0;
                        }
                        if(s.charAt(8+offset)=='S')

f1.write(symSymbol.get(Integer.parseInt(s.substring(10+offset,s.length()-1)))+"\n");
                        else
                                f1.write(litAddr.get(Integer.parseInt(s.substring(10+offset,s.length()-
1)))+"\n");
                }
                 else if(s.substring(1,6).compareToIgnoreCase("DL,01")==0){
                        String s1=s.substring(10,s.length()-1),s2="";
                        for(int i=0;i<3-s1.length();i++)
                                s2+="0";
                        s2+=s1;
                        f1.write("+00 0 "+s2+"\n");
                }
                 else{
                        f1.write("\n");
                }
            }
            f1.close();
            b1.close();
            b2.close();
            b3.close();
        }
}

/*
OUTPUT:
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A2$ javac Pass2.java
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A2$ java Pass2
neha@neha-1011PX:~/Desktop/neha_SPOS/Turn1/A2$ cat Pass2.txt

intermediate code -
(AD,01)(C,200)
(IS,04)(1)(L,1)
(IS,05)(1)(S,1)
(IS,04)(1)(S,1)
(IS,04)(3)(S,3)
(IS,01)(3)(L,2)
```

(IS,07)(6)(S,4)
(DL,01)(C,5)
(DL,01)(C,1)
(IS,02)(1)(L,3)
(IS,07)(1)(S,5)
(IS,00)
(AD,03)(S,2)+2
(IS,03)(3)(S,3)
(AD,03)(S,6)+1
(DL,02)(C,1)
(DL,02)(C,1)
(AD,02)
(DL,01)(C,1)

Symbol Table --

A	211	1
LOOP	202	1
B	212	1
NEXT	208	1
BACK	202	1
LAST	210	1

literal table --

5	206
1	207
1	213

2.

```
import java.io.*;
class SymTab
{
	public static void main(String args[])throws Exception
	{
		FileReader FP=new FileReader(args[0]);
		BufferedReader bufferedReader = new BufferedReader(FP);

		String line=null;
		int line_count=0,LC=0,symTabLine=0,opTabLine=0,litTabLine=0,poolTabLine=0;

		 //Data Structures
		 final int MAX=100;
		 String SymbolTab[][]=new String[MAX][3];
```

```
                String OpTab[][]=new String[MAX][3];
                String LitTab[][]=new String[MAX][2];
                int PoolTab[]=new int[MAX];
                int litTabAddress=0;
/*-----------------------------------------------------------------------------------*/

                System.out.println("_____");
                  while((line=bufferedReader.readLine())!=null)
                   {
                        String[] tokens=line.split("\t");
                        if(line_count==0)
                        {
                                LC=Integer.parseInt(tokens[2]);
                                //set LC to operand of START
                                for(int i=0;i<tokens.length;i++)          //for printing the input program
                                        System.out.print(tokens[i]+"\t");
                                System.out.println("");
                        }
                        else
                        {
                                for(int i=0;i<tokens.length;i++) //for printing the input program
                                        System.out.print(tokens[i]+"\t");
                                System.out.println("");
                                if(!tokens[0].equals(""))
                                {

                                        //Inserting into Symbol Table
                                        SymbolTab[symTabLine][0]=tokens[0];
                                        SymbolTab[symTabLine][1]=Integer.toString(LC);
                                        SymbolTab[symTabLine][2]=Integer.toString(1);
                                        symTabLine++;
                                }
                                else
if(tokens[1].equalsIgnoreCase("DS")||tokens[1].equalsIgnoreCase("DC"))
                                {
                                        //Entry into symbol table for declarative statements
                                        SymbolTab[symTabLine][0]=tokens[0];
                                        SymbolTab[symTabLine][1]=Integer.toString(LC);
                                        SymbolTab[symTabLine][2]=Integer.toString(1);
                                        symTabLine++;
                                }
```

```
                    if(tokens.length==3 && tokens[2].charAt(0)=='=')
                    {
                        //Entry of literals into literal table
                        LitTab[litTabLine][0]=tokens[2];
                        LitTab[litTabLine][1]=Integer.toString(LC);
                        litTabLine++;
                    }

                    else if(tokens[1]!=null)
                    {
                            //Entry of Mnemonic in opcode table
                        OpTab[opTabLine][0]=tokens[1];


        if(tokens[1].equalsIgnoreCase("START")||tokens[1].equalsIgnoreCase("END")||tokens[1].equals
IgnoreCase("ORIGIN")||tokens[1].equalsIgnoreCase("EQU")||tokens[1].equalsIgnoreCase("LTORG"))
        //if Assembler Directive
                        {
                            OpTab[opTabLine][1]="AD";
                            OpTab[opTabLine][2]="R11";

                        }
                        else
if(tokens[1].equalsIgnoreCase("DS")||tokens[1].equalsIgnoreCase("DC"))
                        {
                            OpTab[opTabLine][1]="DL";
                            OpTab[opTabLine][2]="R7";

                        }
                        else
                        {
                            OpTab[opTabLine][1]="IS";
                            OpTab[opTabLine][2]="(04,1)";
                        }
                    opTabLine++;
                    }
                }
                line_count++;
                LC++;
            }
```

```
System.out.println("_____");

                //print symbol table
                System.out.println("\n\n          SYMBOL TABLE          ");
                System.out.println("----------------------");
                System.out.println("SYMBOL\tADDRESS\tLENGTH");
                System.out.println("----------------------");
                for(int i=0;i<symTabLine;i++)

System.out.println(SymbolTab[i][0]+"\t"+SymbolTab[i][1]+"\t"+SymbolTab[i][2]);
                System.out.println("----------------------");


                //print opcode table
                System.out.println("\n\n          OPCODE TABLE          ");
                System.out.println("------------------------");
                System.out.println("MNEMONIC\tCLASS\tINFO");
                System.out.println("------------------------");
                for(int i=0;i<opTabLine;i++)
                        System.out.println(OpTab[i][0]+"\t\t"+OpTab[i][1]+"\t"+OpTab[i][2]);
                System.out.println("------------------------");

                //print literal table
                System.out.println("\n\n  LITERAL TABLE                 ");
                System.out.println("----------------");
                System.out.println("LITERAL\tADDRESS");
                System.out.println("----------------");
                for(int i=0;i<litTabLine;i++)
                        System.out.println(LitTab[i][0]+"\t"+LitTab[i][1]);
                System.out.println("----------------");


                //intialization of POOLTAB
                for(int i=0;i<litTabLine;i++)
                {
                        if(LitTab[i][0]!=null && LitTab[i+1][0]!=null ) //if literals are present
                        {
                                if(i==0)
                                {
                                        PoolTab[poolTabLine]=i+1;
                                        poolTabLine++;
```

```
                                        }
                                        else
if(Integer.parseInt(LitTab[i][1])<(Integer.parseInt(LitTab[i+1][1]))-1)
                                        {
                                                PoolTab[poolTabLine]=i+2;
                                                poolTabLine++;
                                        }
                                }
                        }
                        //print pool table
                        System.out.println("\n\n  POOL TABLE              ");
                        System.out.println("----------------");
                        System.out.println("LITERAL NUMBER");
                        System.out.println("----------------");
                        for(int i=0;i<poolTabLine;i++)
                                System.out.println(PoolTab[i]);
                        System.out.println("----------------");


                // Always close files.
                bufferedReader.close();
        }
}

/*
OUTPUT-
neha@neha-1011PX:~/neha_SPOS$ javac SymTab.java
neha@neha-1011PX:~/neha_SPOS$ java SymTab input.txt

3. fcfs

#include<stdio.h>
int main()
{
int bt[20], wt[20], tat[20], i, n;
float wtavg, tatavg;

printf("\nEnter the number of processes -- ");
scanf("%d", &n);

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

```
printf("\nEnter Burst Time for Process %d -- ", i);
scanf("%d", &bt[i]);
}
wt[0] = wtavg = 0;
tat[0] = tatavg = bt[0];

for(i=1; i<n; i++)
{
wt[i] = wt[i-1] + bt[i-1];
tat[i] = tat[i-1] + bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}

printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0; i<n; i++){
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);}

printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);

printf("\nPress any key to exit...");
getchar();
return 0;
}
```

4. SJF

```
#include <iostream>
#include <algorithm>
#include <iomanip>
#include <string.h>
using namespace std;

struct process {
    int pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
```

```
    int response_time;
};

int main() {

    int n;
    struct process p[100];
    float avg_turnaround_time;
    float avg_waiting_time;
    float avg_response_time;
    float cpu_utilisation;
    int total_turnaround_time  = 0;
    int total_waiting_time = 0;
    int total_response_time = 0;
    int total_idle_time = 0;
    float throughput;
    int burst_remaining[100];
    int is_completed[100];
    memset(is_completed,0,sizeof(is_completed));

    cout << setprecision(2) << fixed;

    cout<<"Enter the number of processes: ";
    cin>>n;

    for(int i = 0; i < n; i++) {
        cout<<"Enter arrival time of process "<<i+1<<": ";
        cin>>p[i].arrival_time;
        cout<<"Enter burst time of process "<<i+1<<": ";
        cin>>p[i].burst_time;
        p[i].pid = i+1;
        burst_remaining[i] = p[i].burst_time;
        cout<<endl;
    }

    int current_time  = 0;
    int completed = 0;
    int prev = 0;

    while(completed != n) {
        int idx = -1;
        int mn = 10000000;
        for(int i = 0; i < n; i++) {
            if(p[i].arrival_time <= current_time && is_completed[i] == 0) {
                if(burst_remaining[i] < mn) {
                    mn = burst_remaining[i];
```

```
                idx = i;
            }
            if(burst_remaining[i] == mn) {
                if(p[i].arrival_time < p[idx].arrival_time) {
                    mn = burst_remaining[i];
                    idx = i;
                }
            }
        }
    }

    if(idx != -1) {
        if(burst_remaining[idx] == p[idx].burst_time) {
            p[idx].start_time = current_time;
            total_idle_time += p[idx].start_time - prev;
        }
        burst_remaining[idx] -= 1;
        current_time++;
        prev = current_time;

        if(burst_remaining[idx] == 0) {
            p[idx].completion_time = current_time;
            p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
            p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
            p[idx].response_time = p[idx].start_time - p[idx].arrival_time;

            total_turnaround_time += p[idx].turnaround_time;
            total_waiting_time += p[idx].waiting_time;
            total_response_time += p[idx].response_time;

            is_completed[idx] = 1;
            completed++;
        }
    }
    else {
         current_time++;
    }
}

int min_arrival_time = 10000000;
int max_completion_time = -1;
for(int i = 0; i < n; i++) {
    min_arrival_time = min(min_arrival_time,p[i].arrival_time);
    max_completion_time = max(max_completion_time,p[i].completion_time);
}
```

```
    avg_turnaround_time  = (float) total_turnaround_time  / n;
    avg_waiting_time = (float) total_waiting_time / n;
    avg_response_time = (float) total_response_time / n;
    cpu_utilisation = ((max_completion_time  - total_idle_time) / (float)
max_completion_time )*100;
    throughput  = float(n) / (max_completion_time  - min_arrival_time);

    cout<<endl<<endl;

    cout<<"#P\t"<<"AT\t"<<"BT\t"<<"ST\t"<<"CT\t"<<"TAT\t"<<"WT\t"<<"RT\t"<<"\n"<<en
dl;

    for(int i = 0; i < n; i++) {
        cout<<p[i].pid<<"\t"<<p[i].arrival_time<<"\t"<<p[i].burst_time<<"\t"<<p[i].start_time<
<"\t"<<p[i].completion_time<<"\t"<<p[i].turnaround_time<<"\t"<<p[i].waiting_time<<"\t"<<
p[i].response_time<<"\t"<<"\n"<<endl;
    }
    cout<<"Average Turnaround  Time = "<<avg_turnaround_time<<endl;
    cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;

}
```

5.priority

```
#include<iostream>
#include<limits>
using namespace std;

class Process{
    public:
        string processName;
        int arrivalTime;
        int burstTime;
        int priority;

        int remainingTime;

  int responseTime;
        int completionTime;

        int waitingTime;
        int turnAroundTime;

        void initialize(){
         remainingTime = burstTime;
        }
};
```

```
int main(){
    int numOfProcesses;
    cout << "Enter no. of processes: ";
    cin >> numOfProcesses;

    Process processes[numOfProcesses];

    for(int n=0;n<numOfProcesses;n++){
        cout << "\nEnter Process Name for " << (n+1) << ": ";
        cin >> processes[n].processName;
        cout << "Enter Arrival Time for Process " << (n+1) << ": ";
        cin >> processes[n].arrivalTime;
        cout << "Enter Burst Time for Process " << (n+1) << ": ";
        cin >> processes[n].burstTime;
        cout << "Enter Priority for Process " << (n+1) << ": ";
        cin >> processes[n].priority;

        processes[n].initialize();
    }

    cout << "\n" << endl;

    for(int i=0;i<numOfProcesses-1;i++){
        for(int j=i+1;j<numOfProcesses;j++){
            if(processes[j].arrivalTime  < processes[i].arrivalTime){
                Process temp = processes[j];
                processes[j] = processes[i];
                processes[i] = temp;
            }
        }
    }

    int currentTime  = 0;

    while(true){

     int currentHighestPriorityIndex  = -1;
     int currentHighestPriority  = numeric_limits<int>::max();

     bool isAllCompleted = true;

     for(int i=0;i<numOfProcesses;i++){
      if(processes[i].remainingTime > 0){
       isAllCompleted = false;
              if(processes[i].arrivalTime  <= currentTime){
```

```
                if(processes[i].priority < currentHighestPriority){
                    currentHighestPriority = processes[i].priority;
                    currentHighestPriorityIndex = i;
                }
            }

     }
    }

    if(isAllCompleted){
     break;
    }

  processes[currentHighestPriorityIndex].responseTime = currentTime;
        processes[currentHighestPriorityIndex].remainingTime = 0;
        currentTime += processes[currentHighestPriorityIndex].burstTime;
        processes[currentHighestPriorityIndex].completionTime = currentTime;
    }


    int sumResponseTime = 0;
    int sumCompletionTime = 0;
    int sumWaitingTime = 0;
    int sumTurnAroundTime = 0;

    for(int n=0;n<numOfProcesses;n++){
        cout << "\nProcess " << processes[n].processName << ":\n";
        cout << "Response Time: " << processes[n].responseTime << endl;
        cout << "Completion Time: " << processes[n].completionTime << endl;
        processes[n].turnAroundTime = processes[n].completionTime -
processes[n].arrivalTime;
        processes[n].waitingTime = processes[n].turnAroundTime -
processes[n].burstTime;
        cout << "Waiting Time: " << processes[n].waitingTime << endl;
        cout << "Turn Around Time: " << processes[n].turnAroundTime << "\n" << endl;

        sumResponseTime += processes[n].responseTime;
        sumCompletionTime += processes[n].completionTime;
        sumWaitingTime += processes[n].waitingTime;
        sumTurnAroundTime += processes[n].turnAroundTime;
    }

 cout << "\n\nAverage Response Time for " << (numOfProcesses) << " Processes: " <<
(float) sumResponseTime/numOfProcesses;
 cout << "\n\nAverage Completion Time for " << (numOfProcesses) << " Processes: " <<
(float) sumCompletionTime/numOfProcesses;
```

```
    cout << "\n\nAverage Waiting Time for " << (numOfProcesses) << " Processes: " <<
(float) sumWaitingTime/numOfProcesses;
    cout << "\n\nAverage Turn Around Time for " << (numOfProcesses) << " Processes: "
<< (float) sumTurnAroundTime/numOfProcesses;

    return 0;
}
```

6.RR

```
#include<stdio.h>

int main() {
int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;
float awt = 0, att = 0, temp = 0;

printf("Enter the number of processes: ");
scanf("%d", &n);

for(i = 0; i < n; i++) {
printf("Enter Burst Time for Process %d: ", i + 1);
scanf("%d", &bu[i]);
ct[i] = bu[i]; // Copy burst time to ct array for later calculations
}

printf("Enter the size of time slice: ");
scanf("%d", &t);

max = bu[0];
for(i = 1; i < n; i++) {
if(max < bu[i])

max = bu[i];
}

for(j = 0; j < (max / t) + 1; j++) {
for(i = 0; i < n; i++) {
if(bu[i] != 0) {
if(bu[i] <= t) {
tat[i] = temp + bu[i];
temp = temp + bu[i];
bu[i] = 0;
} else {
```

```
bu[i] = bu[i] - t;
temp = temp + t;
}
}
}
}

for(i = 0; i < n; i++) {
wa[i] = tat[i] - ct[i];
att += tat[i];
awt += wa[i];
}

printf("\nAverage Turnaround Time: %.2f", att / n);
printf("\nAverage Waiting Time: %.2f\n", awt / n);
printf("\nPROCESS\t BURST TIME \t WAITING TIME\t TURNAROUND TIME\n");

for(i = 0; i < n; i++) {
printf("%d\t %d\t\t %d\t\t %d\n", i + 1, ct[i], wa[i], tat[i]);
}

return 0;}
```

7.best fit

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
static int bf[max],ff[max];
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
```

```
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i];
if(temp>=0)
if(lowest>temp)
{
ff[i]=j;

lowest=temp;
}
}
}
frag[i]=lowest;
bf[ff[i]]=1;
lowest=10000;
}
printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
for(i=1;i<=nf && ff[i]!=0;i++){
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);}
getch();
}
```

8.first fit

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - Worst Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{

for(j=1;j<=nb;j++)
{
```

```
if(bf[j]!=1)  //if bf[j] is not allocated
{
temp=b[j]-f[i];
if(temp>=0)
if(highest<temp)
{
ff[i]=j;
highest=temp;
}
}
}
frag[i]=highest;
bf[ff[i]]=1;
highest=0;
}
printf("\nFile_no:\tFile_size  :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++){
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);}
getch();
}
```

9. worst fit

```
#include<stdio.h>
#include<conio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - First Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i];
if(temp>=0)
{
ff[i]=j;
```

```
break;
}
}
}
frag[i]=temp;
bf[ff[i]]=1;
}
printf("\nFile_no:\tFile_size  :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
getch();
}
```

10. fifo

```
#include<stdio.h>
#include<conio.h>
main()
{
int i, j, k, f, pf=0, count=0, rs[25], m[10], n;
clrscr();
printf("\n Enter the length of reference string -- ");
scanf("%d",&n);
printf("\n Enter the reference string -- ");
for(i=0;i<n;i++)
scanf("%d",&rs[i]);
printf("\n Enter no. of frames -- ");
scanf("%d",&f);
for(i=0;i<f;i++)
m[i]=-1;
printf("\n The Page Replacement Process is -- \n");
for(i=0;i<n;i++)
{
for(k=0;k<f;k++)
{
if(m[k]==rs[i])
break;
}
if(k==f)
{
m[count++]=rs[i];
pf++;
}
for(j=0;j<f;j++)
printf("\t%d",m[j]);
if(k==f)
```

```
printf("\tPF No. %d",pf);
printf("\n");
if(count==f)
count=0;
}
printf("\n The number of Page Faults using FIFO are %d",pf);
getch();
}
```

11.lru

```
#include<stdio.h>
#include<conio.h>
main()
{
int i, j , k, min, rs[25], m[10], count[10], flag[25], n, f, pf=0, next=1;
clrscr();
printf("Enter the length of reference string -- ");
scanf("%d",&n);
printf("Enter the reference string -- ");
for(i=0;i<n;i++)
{
scanf("%d",&rs[i]);
flag[i]=0;
}
printf("Enter the number of frames -- ");
scanf("%d",&f);
for(i=0;i<f;i++)
{
count[i]=0;
m[i]=-1;
}
printf("\nThe Page Replacement process is -- \n");
for(i=0;i<n;i++)
{
for(j=0;j<f;j++)
{
if(m[j]==rs[i])
{
flag[i]=1;
count[j]=next;
next++;
}
```

```
}
if(flag[i]==0)
{
if(i<f)
{
m[i]=rs[i];

count[i]=next;
next++;
}
else
{
min=0;
for(j=1;j<f;j++)
if(count[min] >count[j])
min=j;
m[min]=rs[i];
count[min]=next;
next++;
}
pf++;
}
for(j=0;j<f;j++)
printf("%d\t", m[j]);
if(flag[i]==0)
printf("PF No. -- %d" , pf);
printf("\n");
}
printf("\nThe number of page faults using LRU are %d",pf);
getch();
}
```

12.lfu

```
#include<stdio.h>
#include<conio.h>
main()
{
int rs[50], i, j, k, m, f, cntr[20], a[20], min, pf=0;
clrscr();
printf("\nEnter number of page references -- ");
scanf("%d",&m);
```

```
printf("\nEnter the reference string -- ");
for(i=0;i<m;i++)
scanf("%d",&rs[i]);
printf("\nEnter the available no. of frames -- ");
scanf("%d",&f);
for(i=0;i<f;i++)
{
cntr[i]=0;
a[i]=-1;
}
Printf("\nThe Page Replacement Process is –\n");
for(i=0;i<m;i++)
{
for(j=0;j<f;j++)
if(rs[i]==a[j])
{
cntr[j]++;
break;
}
if(j==f)
{
min = 0;
for(k=1;k<f;k++)
if(cntr[k]<cntr[min])
min=k;
a[min]=rs[i];
cntr[min]=1;
pf++;
}
printf("\n");
for(j=0;j<f;j++)
printf("\t%d",a[j]);
if(j==f)
printf("\tPF No. %d",pf);
}
printf("\n\n Total number of page faults -- %d",pf);
getch();
}
```