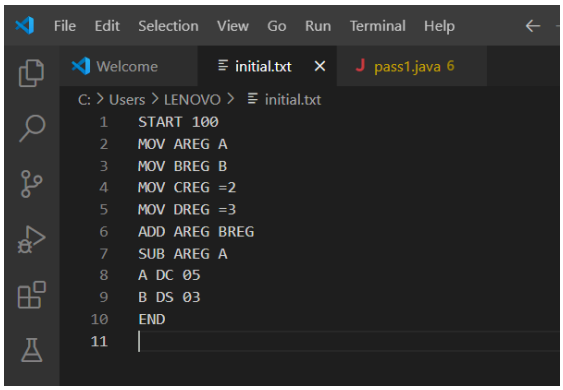
Assignment :- 1

```
import java.util.*;
import java.io.*;

public class Pass1 {
    static int address = 0;
    static int sadd[] = new int[10];
    static int ladd[] = new int[10];

    public static void main(String args[]) {
        BufferedReader br;
        OutputStream oo;
        String input = null;

        String IS[] = {"ADD", "SUB", "MUL", "MOV"};
        String UserReg[] = {"AREG", "BREG", "CREG", "DREG"};
        String AD[] = {"START", "END"};
        String DL[] = {"DC", "DS"};

        int lc = 0;
        int scount = 0, lcount = 0;
        int flag = 0, flag2 = 0, stored = 0;

        String tokens[] = new String[30];
        String tt = null;
        String sv[] = new String[10];
        String lv[] = new String[10];

        try {
            br = new BufferedReader(new FileReader("initial.txt"));
            File f = new File("IM.txt");
            File f1 = new File("ST.txt");
            File f2 = new File("LT.txt");
            PrintWriter p = new PrintWriter(f);
            PrintWriter p1 = new PrintWriter(f1);
            PrintWriter p2 = new PrintWriter(f2);

            int k = 0, l = 0;

            while ((input = br.readLine()) != null) {
                StringTokenizer st = new StringTokenizer(input, " ");
                while (st.hasMoreTokens()) {
                    tt = st.nextToken();

                    if (tt.matches("\\d*") && tt.length() > 2) {
                        lc = Integer.parseInt(tt);
                        p.println(lc);
                        address = lc - 1;
                    } else {
                        for (int i = 0; i < AD.length; i++) {
                            if (tt.equals(AD[i])) {
                                p.print("AD " + (i + 1) + " ");
```

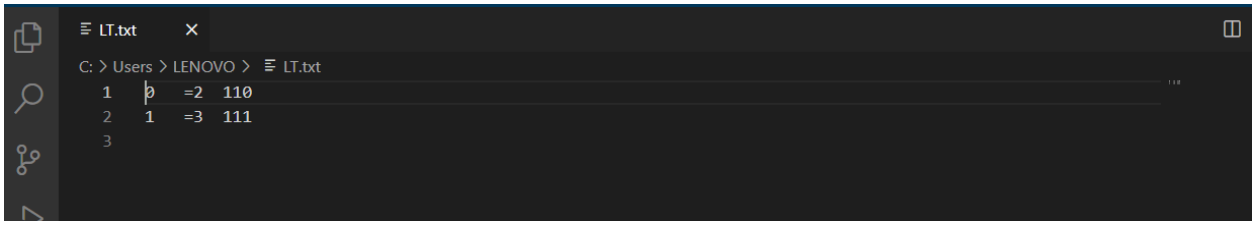
```
    }
}

for (int i = 0; i < IS.length; i++) {
    if (tt.equals(IS[i])) {
        p.print("IS " + (i + 1) + " ");
    }
}

for (int i = 0; i < UserReg.length; i++) {
    if (tt.equals(UserReg[i])) {
        p.print((i + 1) + " ");
        flag = 1;
    }
}

for (int i = 0; i < DL.length; i++) {
    if (tt.equals(DL[i])) {
        p.print("DL " + (i + 1) + " ");
    }
}

if (tt.length() == 1 && !st.hasMoreTokens() && flag == 1) {
    if (Arrays.asList(sv).contains(tt)) {
        for (int i = 0; i < scount; i++) {
            if (sv[i].equals(tt)) {
                p.print("S" + i);
                flag2 = 1;
            } else {
                flag2 = 0;
            }
        }
    } else {
        p.print("S" + scount);
        sv[scount] = tt;
        flag2 = 1;
        scount++;
    }
}

if (tt.length() == 1 && st.hasMoreTokens()) {
    p.print(tt + " ");
    sadd[k] = address;
    k++;
}

if (tt.charAt(0) == '=') {
    p.print("L" + lcount);
    lv[lcount] = tt;
    lcount++;
}

if (!st.hasMoreTokens()) {
    p.println();
```

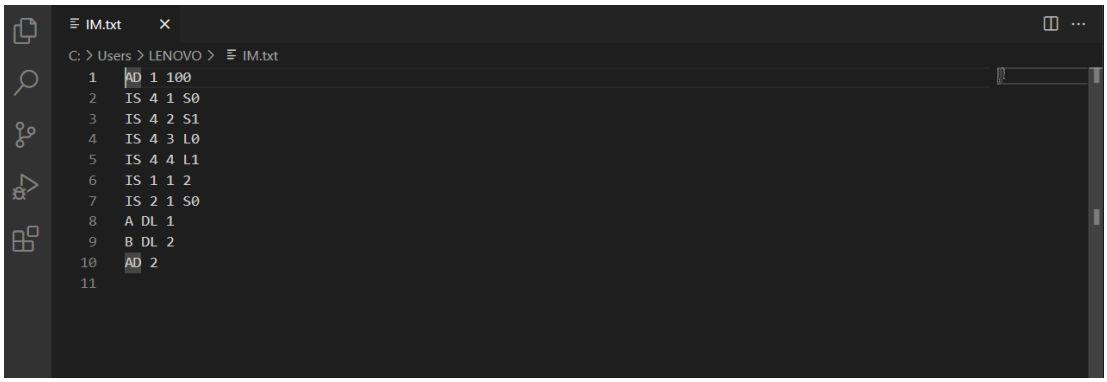
```
                }

                if (tt.equals("DS")) {
                    int a = Integer.parseInt(st.nextToken());
                    address = address + a - 1;
                    p.println();
                }
            }
        }
        address++;
    }

    p.close();
    address--;

    for (int i = 0; i < lcount; i++) {
        ladd[i] = address;
        address++;
    }

    for (int i = 0; i < scount; i++) {
        p1.println(i + "\t" + sv[i] + "\t" + sadd[i]);
    }
    p1.close();

    for (int i = 0; i < lcount; i++) {
        p2.println(i + "\t" + lv[i] + "\t" + ladd[i]);
    }
    p2.close();

  } catch (Exception e) {
    e.printStackTrace();
  }
 }
}
```

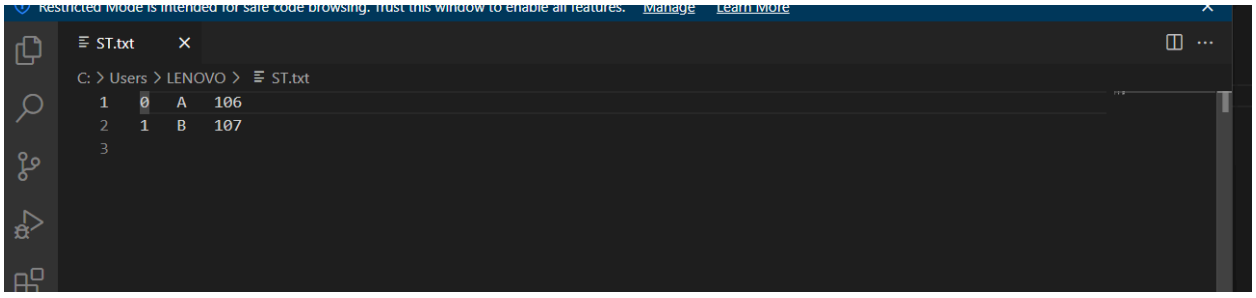
Initial :-



OUTPUT :-







Assignment :- 2

Code :-

```
import java.util.*;
import java.io.*;
public class macro
{
	public static void main(String args[])
	{

		BufferedReader br;
		OutputStream oo;
		String input=null;
		String tt=null;
		String arg=null;
		String macroTokens=null;
		String mnt[]=new String[10];
		String mdt[]=new String[20];
		String AR[]=new String[20];
		int macroindex[]=new int[10];
		int mcount=0,arg_count=0;
		int middlecount=0;
		int index=1;
		int macro_enc=0;

		try
		{
			br=new BufferedReader(new FileReader("Input.txt"));
			File f3 = new File("mnt.txt");
			File f4 = new File("mdt.txt");
			File f5 = new File("adt.txt");
			PrintWriter p3 = new PrintWriter(f3);
			PrintWriter p4 = new PrintWriter(f4);
					PrintWriter p5 = new PrintWriter(f5);
			while ((input = br.readLine()) != null)
			{
				StringTokenizer st = new StringTokenizer(input," ");
			  tt=st.nextToken();
			  if(tt.equals("MACRO"))
			  {
					macro_enc=1;
					tt=st.nextToken();
					mnt[mcount]=tt;
					macroindex[mcount]=index;
					p3.println(mnt[mcount]+"\t"+macroindex[mcount]);
					p4.println(mnt[mcount]);
					p5.println(mnt[mcount]);
					mcount++;

					tt=st.nextToken();
					StringTokenizer t = new StringTokenizer(tt,",");
					while (t.hasMoreTokens())
					{
						arg=t.nextToken();
						if(arg.charAt(0)=='&')
						    {
						      AR[arg_count]=arg;
```

```
                                        p5.println(AR[arg_count]);
                                        arg_count++;
                                    }
                                }
                        }

                        else
                        {
                                if(macro_enc==1)
                                {
                                    if(input.equals("MEND"))
                                    {
                                            macro_enc=0;
                                            p4.println("MEND");
                                    }
                                    else
                                    {
                                            StringTokenizer t=new StringTokenizer(input," ");
                                            while(t.hasMoreTokens())
                                            {
                                                macroTokens=t.nextToken();
                                                for(int i=0;i<arg_count;i++)
                                                {
                                                  if(macroTokens.charAt(0)=='&' && macroTokens.equals(AR[i]))
                                                  {
                                                        p4.print("AR"+i);
                                                  }
                                                }
                                                if(macroTokens.charAt(0)=='&'){}
                                                else
                                    {
                                                        p4.print(macroTokens+" ");
                                    }
                                                if(!t.hasMoreTokens())
                                    {
                                                        p4.println();
                                    }


                                            }
                                    }

                                }
                         }
                        index++;
                }
                p3.close();
                p4.close();
                p5.close();



        }
        catch(Exception e)
        {
                e.printStackTrace();
        }
    }

}
```

Input File :-

```
1  MACRO
2  M1       &X, &Y, &A=AREG, &B=
3  MOVER    &A, &X
4  ADD      &A, ='1'
5  MOVER    &B, &Y
6  ADD      &B, ='5'
7  MEND
8  MACRO
9  M2       &P, &Q, &U=CREG, &V=DREG
10 MOVER    &U, &P
11 MOVER    &V, &Q
12 ADD      &U, ='15'
13 ADD      &V, ='10'
14 MEND
15 START    100
16 M1       10, 20, &B=CREG
17 M2       100, 200, &V=AREG, &U=BREG
18 END
```

Output :-

```
Open a file
          (P,3)    (P,1)
          (P,3)    ='1'
3 MOVER   (P,4)    (P,2)
4 ADD     (P,4)    ='5'
5 MEND
6 MOVER   (P,7)    (P,5)
7 MOVER   (P,8)    (P,6)
8 ADD     (P,7)    ='15'
9 ADD     (P,8)    ='10'
10 MEND
```

Macro Definition Table

```
M1      1
M2      6
```

Macro Name Table

```
1 M1:     X       Y       A       B
2 M2:     P       Q       U       V
```

Argument List Array

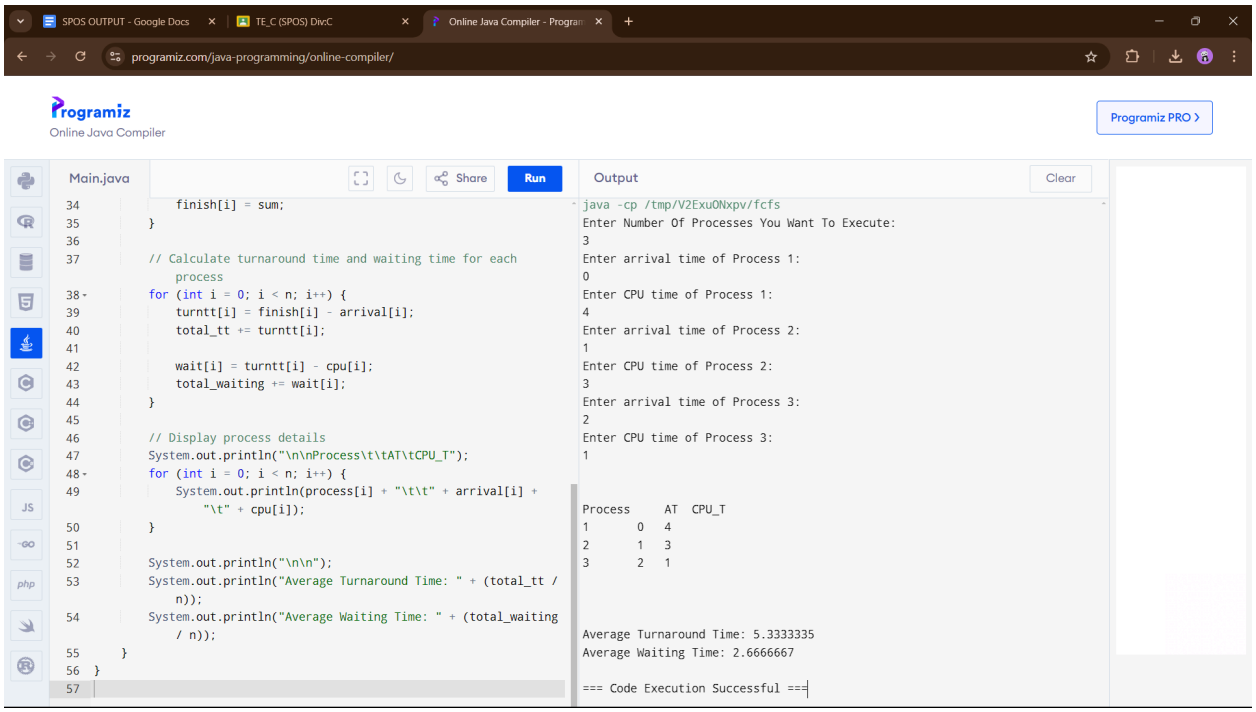
Assignment :- 3

Code:-

```
import java.util.*;
import java.io.*;

public class fcfs {
   public static void main(String args[]) {
      int n, sum = 0;
      float total_tt = 0, total_waiting = 0;
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Number Of Processes You Want To Execute:");
      n = s.nextInt();

      int arrival[] = new int[n];
      int cpu[] = new int[n];
      int finish[] = new int[n];
      int turntt[] = new int[n];
      int wait[] = new int[n];
      int process[] = new int[n];

      // Input arrival times and CPU times for each process
      for (int i = 0; i < n; i++) {
         System.out.println("Enter arrival time of Process " + (i + 1) + ": ");
         arrival[i] = s.nextInt();

         System.out.println("Enter CPU time of Process " + (i + 1) + ": ");
         cpu[i] = s.nextInt();

         process[i] = i + 1;
      }

      // Calculate finish times for each process
      for (int i = 0; i < n; i++) {
         sum += cpu[i];
         finish[i] = sum;
      }

      // Calculate turnaround time and waiting time for each process
      for (int i = 0; i < n; i++) {
         turntt[i] = finish[i] - arrival[i];
```

```
            total_tt += turntt[i];

            wait[i] = turntt[i] - cpu[i];
            total_waiting += wait[i];
        }

        // Display process details
        System.out.println("\n\nProcess\t\tAT\tCPU_T");
        for (int i = 0; i < n; i++) {
            System.out.println(process[i] + "\t\t" + arrival[i] + "\t" + cpu[i]);
        }

        System.out.println("\n\n");
        System.out.println("Average Turnaround Time: " + (total_tt / n));
        System.out.println("Average Waiting Time: " + (total_waiting / n));
    }
}
```

Output :-



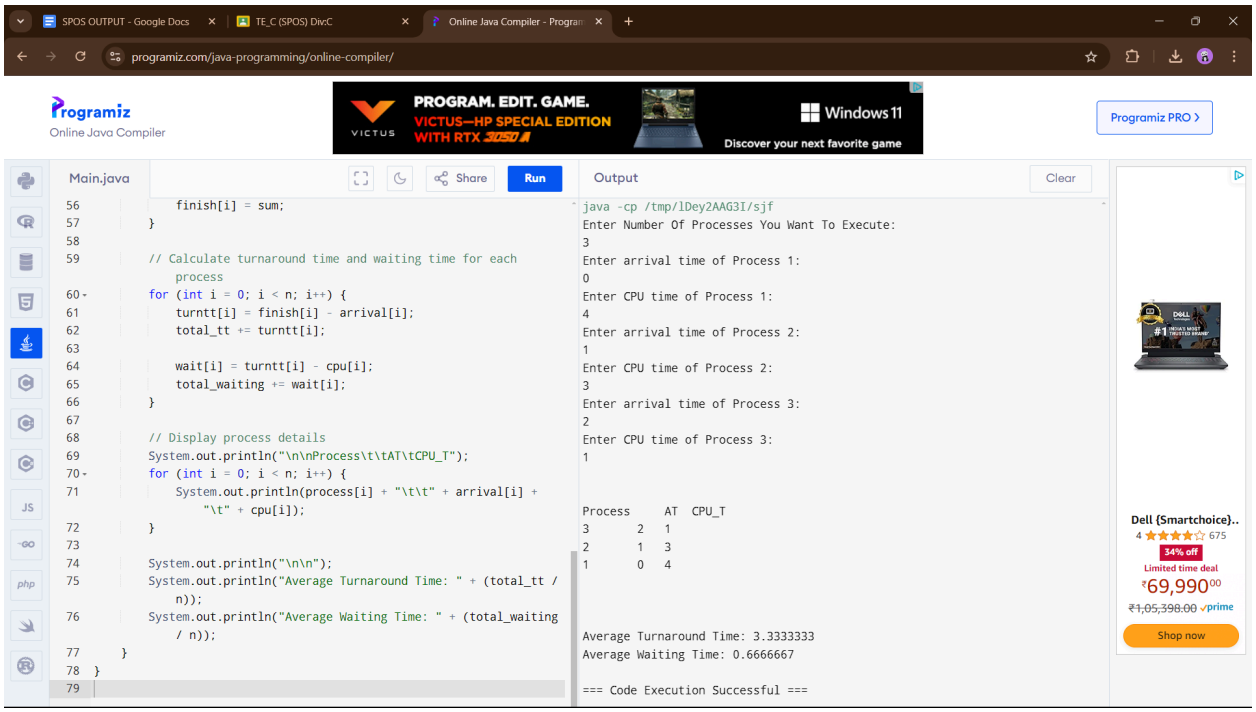
Assignment :- 4

Code:-

```
import java.util.*;
import java.io.*;

public class sjf {
    public static void main(String args[]) {
        int n, sum = 0;
        float total_tt = 0, total_waiting = 0;
        Scanner s = new Scanner(System.in);

        System.out.println("Enter Number Of Processes You Want To Execute:");
        n = s.nextInt();

        int arrival[] = new int[n];
        int cpu[] = new int[n];
        int finish[] = new int[n];
        int turntt[] = new int[n];
        int wait[] = new int[n];
        int process[] = new int[n];

        // Input arrival times and CPU burst times for each process
        for (int i = 0; i < n; i++) {
            System.out.println("Enter arrival time of Process " + (i + 1) + ": ");
            arrival[i] = s.nextInt();

            System.out.println("Enter CPU time of Process " + (i + 1) + ": ");
            cpu[i] = s.nextInt();

            process[i] = i + 1;
        }

        // Sorting processes by CPU burst time using Bubble Sort for SJF scheduling
        for (int i = 0; i < n - 1; i++) {
            for (int j = i + 1; j < n; j++) {
                if (cpu[i] > cpu[j]) {
                    // Swap CPU burst time
                    int temp = cpu[i];
                    cpu[i] = cpu[j];
                    cpu[j] = temp;
```

```
                // Swap arrival time
                temp = arrival[i];
                arrival[i] = arrival[j];
                arrival[j] = temp;

                // Swap process number
                temp = process[i];
                process[i] = process[j];
                process[j] = temp;
            }
        }
    }
    // Calculate finish times
    for (int i = 0; i < n; i++) {
        sum += cpu[i];
        finish[i] = sum;
    }

    // Calculate turnaround time and waiting time for each process
    for (int i = 0; i < n; i++) {
        turntt[i] = finish[i] - arrival[i];
        total_tt += turntt[i];

        wait[i] = turntt[i] - cpu[i];
        total_waiting += wait[i];
    }

    // Display process details
    System.out.println("\n\nProcess\t\tAT\tCPU_T");
    for (int i = 0; i < n; i++) {
        System.out.println(process[i] + "\t\t" + arrival[i] + "\t" + cpu[i]);
    }

    System.out.println("\n\n");
    System.out.println("Average Turnaround Time: " + (total_tt / n));
    System.out.println("Average Waiting Time: " + (total_waiting / n));
  }
}
```

Output :-


SPOS OUTPUT - Google Docs
TE_C (SPOS) DivC
Online Java Compiler - Program
programiz.com/java-programming/online-compiler/
Programiz
Online Java Compiler
PROGRAM. EDIT. GAME.
VICTUS—HP SPECIAL EDITION
WITH RTX 3050 A
Windows 11
Discover your next favorite game
Programiz PRO
Main.java
Share
Run
Output
Clear
56 finish[i] = sum;
57 }
58
59 // Calculate turnaround time and waiting time for each process
60 for (int i = 0; i < n; i++) {
61 turntt[i] = finish[i] - arrival[i];
62 total_tt += turntt[i];
63
64 wait[i] = turntt[i] - cpu[i];
65 total_waiting += wait[i];
66 }
67
68 // Display process details
69 System.out.println("\n\nProcess\t\tAT\tCPU_T");
70 for (int i = 0; i < n; i++) {
71 System.out.println(process[i] + "\t\t" + arrival[i] + "\t" + cpu[i]);
72 }
73
74 System.out.println("\n\n");
75 System.out.println("Average Turnaround Time: " + (total_tt / n));
76 System.out.println("Average Waiting Time: " + (total_waiting / n));
77 }
78 }
79
java -cp /tmp/lDey2AAG3I/sjf
Enter Number Of Processes You Want To Execute:
3
Enter arrival time of Process 1:
0
Enter CPU time of Process 1:
4
Enter arrival time of Process 2:
1
Enter CPU time of Process 2:
3
Enter arrival time of Process 3:
2
Enter CPU time of Process 3:
1
Process AT CPU_T
3 2 1
2 1 3
1 0 4
Average Turnaround Time: 3.3333333
Average Waiting Time: 0.6666667
=== Code Execution Successful ===
Dell {Smartchoice}..
4 675
34% off
Limited time deal
₹69,990
₹1,05,398.00 prime
Shop now

Assignment :- 5

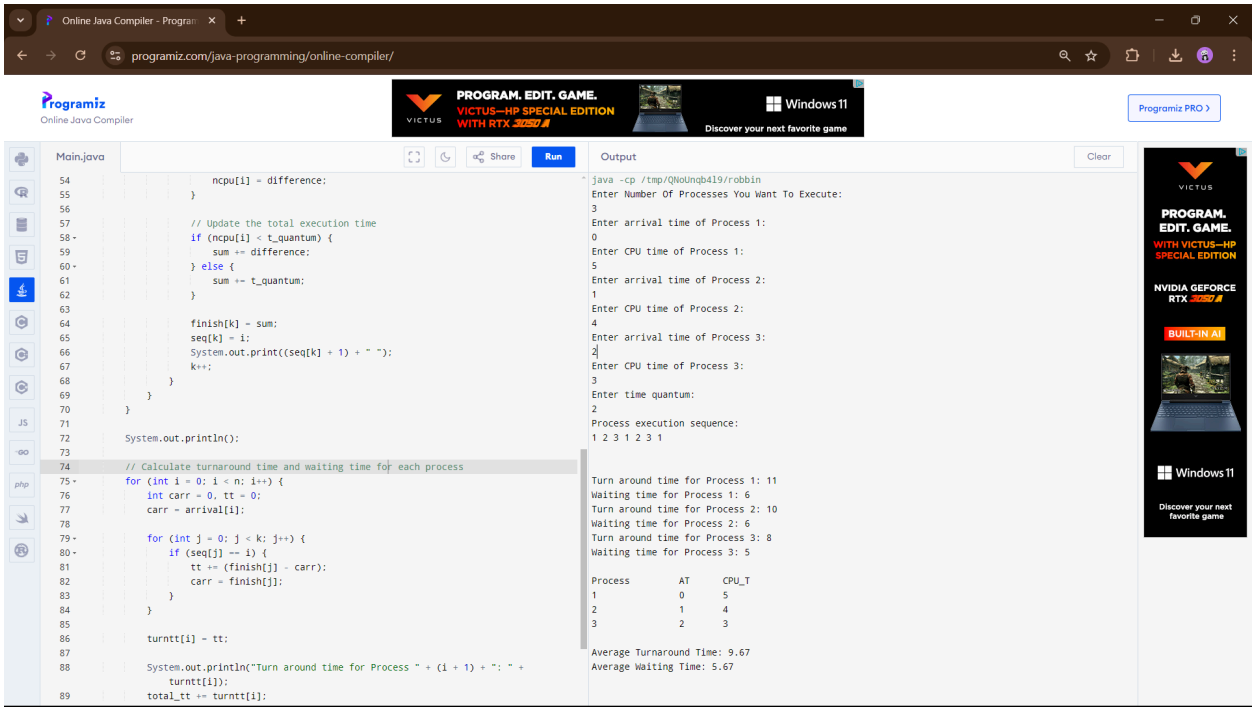
Code :-

```
import java.util.*;
import java.io.*;

public class robbin {
   public static void main(String args[]) {
      int n, sum = 0;
      float total_tt = 0, total_waiting = 0;
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Number Of Processes You Want To Execute:");
      n = s.nextInt();

      int arrival[] = new int[n];
      int cpu[] = new int[n];
      int ncpu[] = new int[n];
      int finish[] = new int[100];
      int turntt[] = new int[n];
      int wait[] = new int[n];
      int process[] = new int[n];
      int seq[] = new int[100];
      int t_quantum, difference, temp_sum = 0, k = 0;

      // Input arrival times and CPU burst times for each process
      for (int i = 0; i < n; i++) {
         System.out.println("Enter arrival time of Process " + (i + 1) + ": ");
         arrival[i] = s.nextInt();

         System.out.println("Enter CPU time of Process " + (i + 1) + ": ");
         ncpu[i] = cpu[i] = s.nextInt();

         process[i] = i + 1;
      }

      // Input the time quantum
      System.out.println("Enter time quantum: ");
      t_quantum = s.nextInt();

      // Calculate total CPU time needed for all processes
```

```
for (int i = 0; i < n; i++) {
    temp_sum += cpu[i];
}

System.out.println("Process execution sequence: ");

// Round Robin scheduling logic
while (sum != temp_sum) {
    for (int i = 0; i < n; i++) {
        if (ncpu[i] > 0) {
            if (ncpu[i] < t_quantum) {
                difference = ncpu[i];
                ncpu[i] = 0;
            } else {
                difference = ncpu[i] - t_quantum;
                ncpu[i] = difference;
            }

            // Update the total execution time
            if (ncpu[i] < t_quantum) {
                sum += difference;
            } else {
                sum += t_quantum;
            }

            finish[k] = sum;
            seq[k] = i;
            System.out.print((seq[k] + 1) + " ");
            k++;
        }
    }
}

System.out.println();

// Calculate turnaround time and waiting time for each process
for (int i = 0; i < n; i++) {
    int carr = 0, tt = 0;
    carr = arrival[i];

    for (int j = 0; j < k; j++) {
        if (seq[j] == i) {
            tt += (finish[j] - carr);
            carr = finish[j];
```

```
            }
        }
        turntt[i] = tt;

        System.out.println("Turn around time for Process " + (i + 1) + ": " + turntt[i]);
        total_tt += turntt[i];

        wait[i] = turntt[i] - cpu[i];
        System.out.println("Waiting time for Process " + (i + 1) + ": " + wait[i]);
        total_waiting += wait[i];
    }

    // Display process details
    System.out.println("\n\nProcess\t\tAT\tCPU_T");
    for (int i = 0; i < n; i++) {
        System.out.println(process[i] + "\t\t" + arrival[i] + "\t" + cpu[i]);
    }
    System.out.println("\n\n");
    System.out.println("Average Turnaround Time: " + (total_tt / n));
    System.out.println("Average Waiting Time: " + (total_waiting / n));
  }
}
```

Output :-



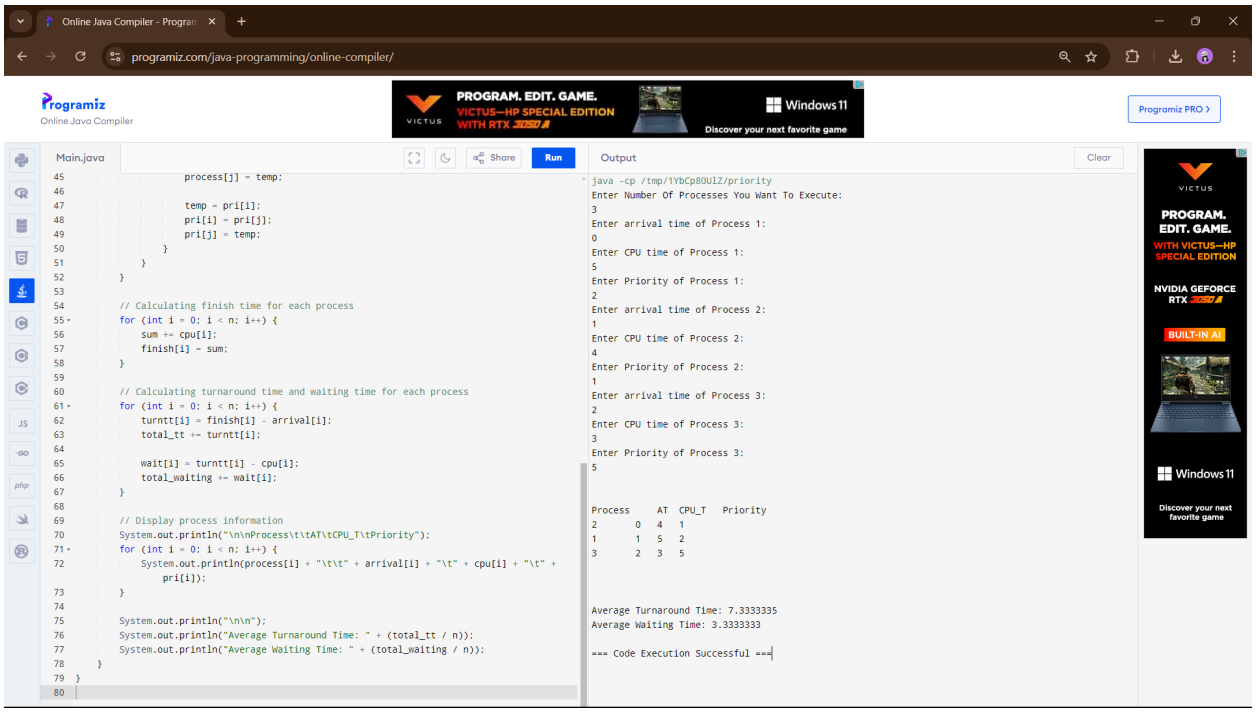
Assignment :- 6

Code:-

```
import java.util.*;
import java.io.*;

public class priority {
    public static void main(String args[]) {
        int n, sum = 0;
        float total_tt = 0, total_waiting = 0;
        Scanner s = new Scanner(System.in);

        System.out.println("Enter Number Of Processes You Want To Execute:");
        n = s.nextInt();

        int arrival[] = new int[n];
        int cpu[] = new int[n];
        int pri[] = new int[n];
        int finish[] = new int[n];
        int turntt[] = new int[n];
        int wait[] = new int[n];
        int process[] = new int[n];

        // Input the arrival time, CPU burst time, and priority for each process
        for (int i = 0; i < n; i++) {
            System.out.println("Enter arrival time of Process " + (i + 1) + ": ");
            arrival[i] = s.nextInt();

            System.out.println("Enter CPU time of Process " + (i + 1) + ": ");
            cpu[i] = s.nextInt();

            System.out.println("Enter Priority of Process " + (i + 1) + ": ");
            pri[i] = s.nextInt();

            process[i] = i + 1;
        }
        // Sorting processes based on priority (lower value indicates higher priority)
        for (int i = 0; i < n - 1; i++) {
            for (int j = i + 1; j < n; j++) {
                if (pri[i] > pri[j]) {
                    int temp = cpu[i];
                    cpu[i] = cpu[j];
```

```
                cpu[j] = temp;

                temp = process[i];
                process[i] = process[j];
                process[j] = temp;

                temp = pri[i];
                pri[i] = pri[j];
                pri[j] = temp;
            }
        }
    }
    // Calculating finish time for each process
    for (int i = 0; i < n; i++) {
        sum += cpu[i];
        finish[i] = sum;
    }
    // Calculating turnaround time and waiting time for each process
    for (int i = 0; i < n; i++) {
        turntt[i] = finish[i] - arrival[i];
        total_tt += turntt[i];

        wait[i] = turntt[i] - cpu[i];
        total_waiting += wait[i];
    }

    // Display process information
    System.out.println("\n\nProcess\t\tAT\tCPU_T\tPriority");
    for (int i = 0; i < n; i++) {
        System.out.println(process[i] + "\t\t" + arrival[i] + "\t" + cpu[i] + "\t" + pri[i]);
    }

    System.out.println("\n\n");
    System.out.println("Average Turnaround Time: " + (total_tt / n));
    System.out.println("Average Waiting Time: " + (total_waiting / n));
  }
}
```

Output :-


Online Java Compiler - Program
programiz.com/java-programming/online-compiler/
Programiz
Online Java Compiler
PROGRAM. EDIT. GAME.
VICTUS—HP SPECIAL EDITION
WITH RTX 3050
Windows 11
Discover your next favorite game
Programiz PRO
Main.java
Share
Run
Output
Clear
process[j] = temp;
temp = pri[i];
pri[i] = pri[j];
pri[j] = temp;
// Calculating finish time for each process
for (int i = 0; i < n; i++) {
sum += cpu[i];
finish[i] = sum;
// Calculating turnaround time and waiting time for each process
for (int i = 0; i < n; i++) {
turntt[i] = finish[i] - arrival[i];
total_tt += turntt[i];
wait[i] = turntt[i] - cpu[i];
total_waiting += wait[i];
// Display process information
System.out.println("\n\nProcess\t\tAT\tCPU_T\tPriority");
for (int i = 0; i < n; i++) {
System.out.println(process[i] + "\t\t" + arrival[i] + "\t" + cpu[i] + "\t" +
pri[i]);
System.out.println("\n\n");
System.out.println("Average Turnaround Time: " + (total_tt / n));
System.out.println("Average Waiting Time: " + (total_waiting / n));
}
}
java -cp /tmp/1YbCp8OUlZ/priority
Enter Number Of Processes You Want To Execute:
3
Enter arrival time of Process 1:
0
Enter CPU time of Process 1:
5
Enter Priority of Process 1:
2
Enter arrival time of Process 2:
1
Enter CPU time of Process 2:
4
Enter Priority of Process 2:
1
Enter arrival time of Process 3:
2
Enter CPU time of Process 3:
3
Enter Priority of Process 3:
5
Process AT CPU_T Priority
2 0 4 1
1 1 5 2
3 2 3 5
Average Turnaround Time: 7.3333335
Average Waiting Time: 3.3333333
=== Code Execution Successful ===
VICTUS
PROGRAM. EDIT. GAME.
WITH VICTUS—HP SPECIAL EDITION
NVIDIA GEFORCE RTX 3050
BUILT-IN AI
Windows 11
Discover your next favorite game

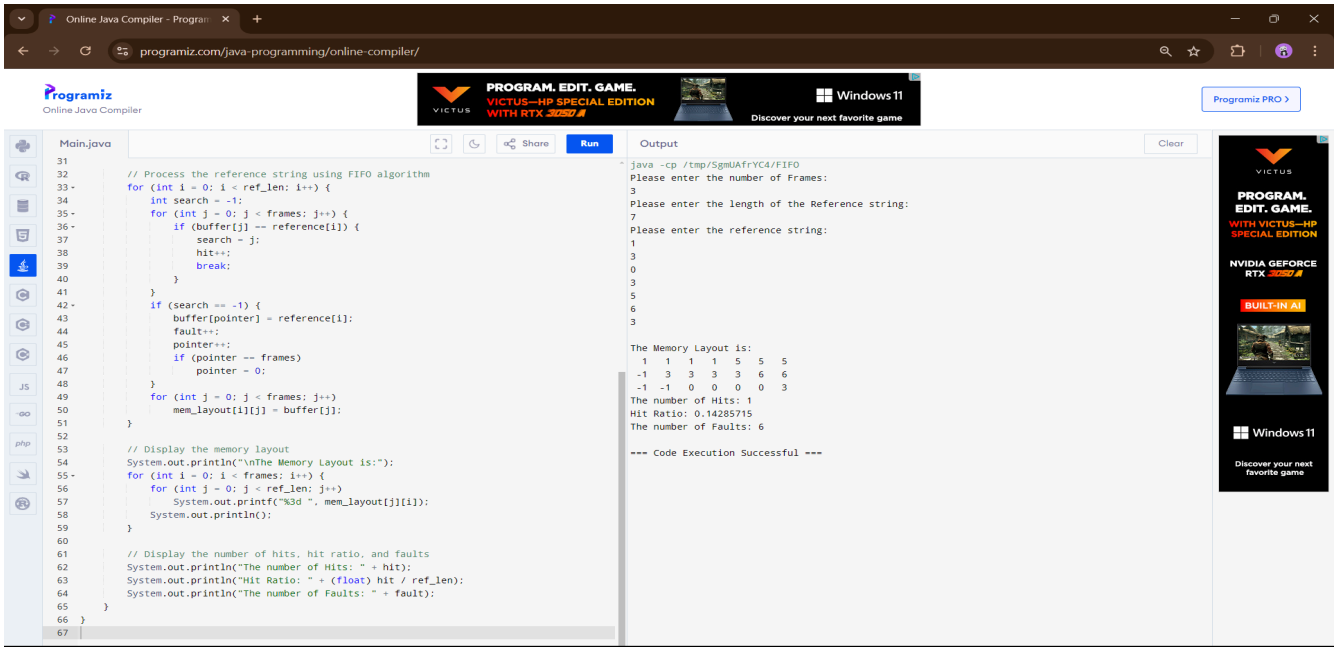
Assignment :- 7

Code :-

```
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_layout[][];

      // Input the number of frames
      System.out.println("Please enter the number of Frames: ");
      frames = Integer.parseInt(br.readLine());

      // Input the length of the reference string
      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;
      // Input the reference string
      System.out.println("Please enter the reference string: ");
      for (int i = 0; i < ref_len; i++) {
         reference[i] = Integer.parseInt(br.readLine());
      }
      // Process the reference string using FIFO algorithm
      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];
    }
    // Display the memory layout
    System.out.println("\nThe Memory Layout is:");
    for (int i = 0; i < frames; i++) {
        for (int j = 0; j < ref_len; j++)
            System.out.printf("%3d ", mem_layout[j][i]);
        System.out.println();
    }
    // Display the number of hits, hit ratio, and faults
    System.out.println("The number of Hits: " + hit);
    System.out.println("Hit Ratio: " + (float) hit / ref_len);
    System.out.println("The number of Faults: " + fault);
  }
}
```

Output :-



Assignment :- 8

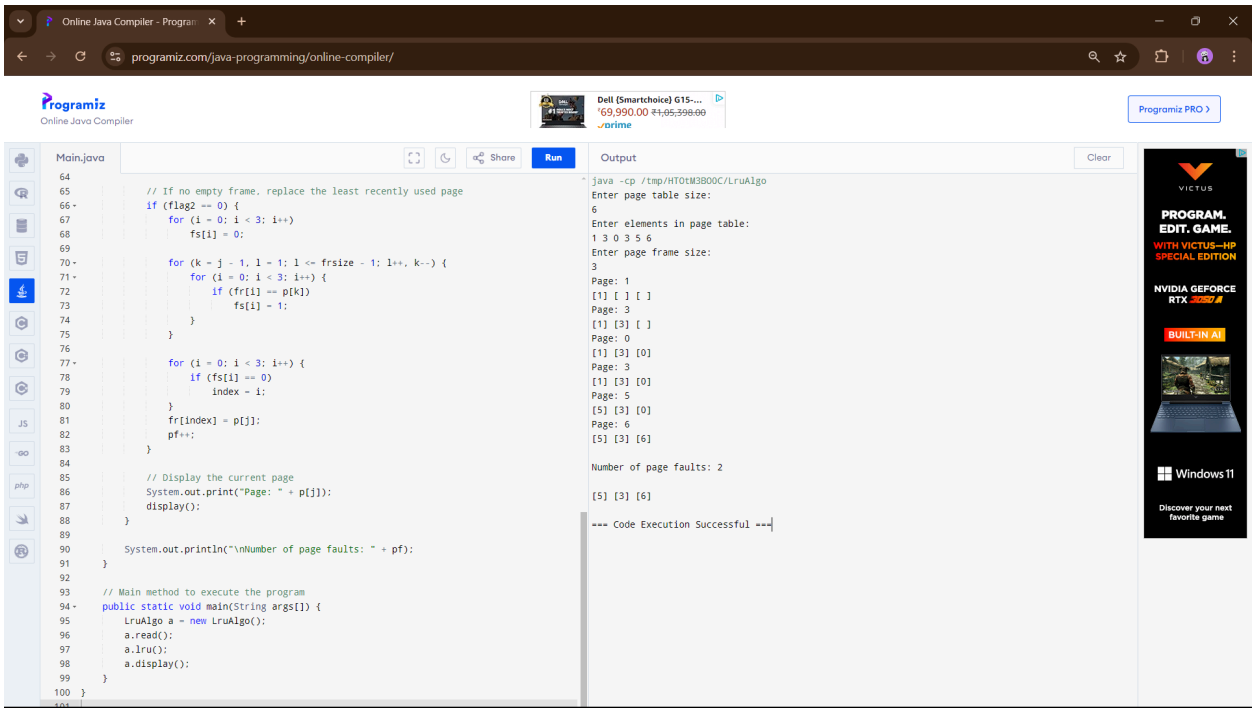
Code:-

```
import java.util.*;

class LruAlgo {
    int p[], n, fr[], m, fs[], index, k, l, flag1 = 0, flag2 = 0, pf = 0, frsize = 3, i, j;
    Scanner src = new Scanner(System.in);

    // Method to read the page table and frame size
    void read() {
        System.out.println("Enter page table size:");
        n = src.nextInt();
        p = new int[n];
        System.out.println("Enter elements in page table:");
        for (int i = 0; i < n; i++)
            p[i] = src.nextInt();
        System.out.println("Enter page frame size:");
        m = src.nextInt();
        fr = new int[m];
        fs = new int[m];
    }

    // Method to display the current frame
    void display() {
        System.out.println();
        for (i = 0; i < m; i++) {
            if (fr[i] == -1)
                System.out.print("[ ] ");
            else
                System.out.print("[" + fr[i] + "] ");
        }
        System.out.println();
    }

    // Method to implement the LRU page replacement algorithm
    void lru() {
        // Initialize the frame array with -1
        for (i = 0; i < m; i++) {
            fr[i] = -1;
        }

        // Start processing the page table
```

```
for (j = 0; j < n; j++) {
    flag1 = 0;
    flag2 = 0;

    // Check if the page is already in the frame
    for (i = 0; i < m; i++) {
        if (fr[i] == p[j]) {
            flag1 = 1;
            flag2 = 1;
            break;
        }
    }

    // Check for an empty frame
    if (flag1 == 0) {
        for (i = 0; i < m; i++) {
            if (fr[i] == -1) {
                fr[i] = p[j];
                flag2 = 1;
                break;
            }
        }
    }

    // If no empty frame, replace the least recently used page
    if (flag2 == 0) {
        for (i = 0; i < 3; i++)
            fs[i] = 0;

        for (k = j - 1, l = 1; l <= frsize - 1; l++, k--) {
            for (i = 0; i < 3; i++) {
                if (fr[i] == p[k])
                    fs[i] = 1;
            }
        }

        for (i = 0; i < 3; i++) {
            if (fs[i] == 0)
                index = i;
        }
        fr[index] = p[j];
        pf++;
    }
```

```
            // Display the current page
            System.out.print("Page: " + p[j]);
            display();
        }

        System.out.println("\nNumber of page faults: " + pf);
    }

    // Main method to execute the program
    public static void main(String args[]) {
        LruAlgo a = new LruAlgo();
        a.read();
        a.lru();
        a.display();
    }
}
```

Output :-



Assignment :- 9

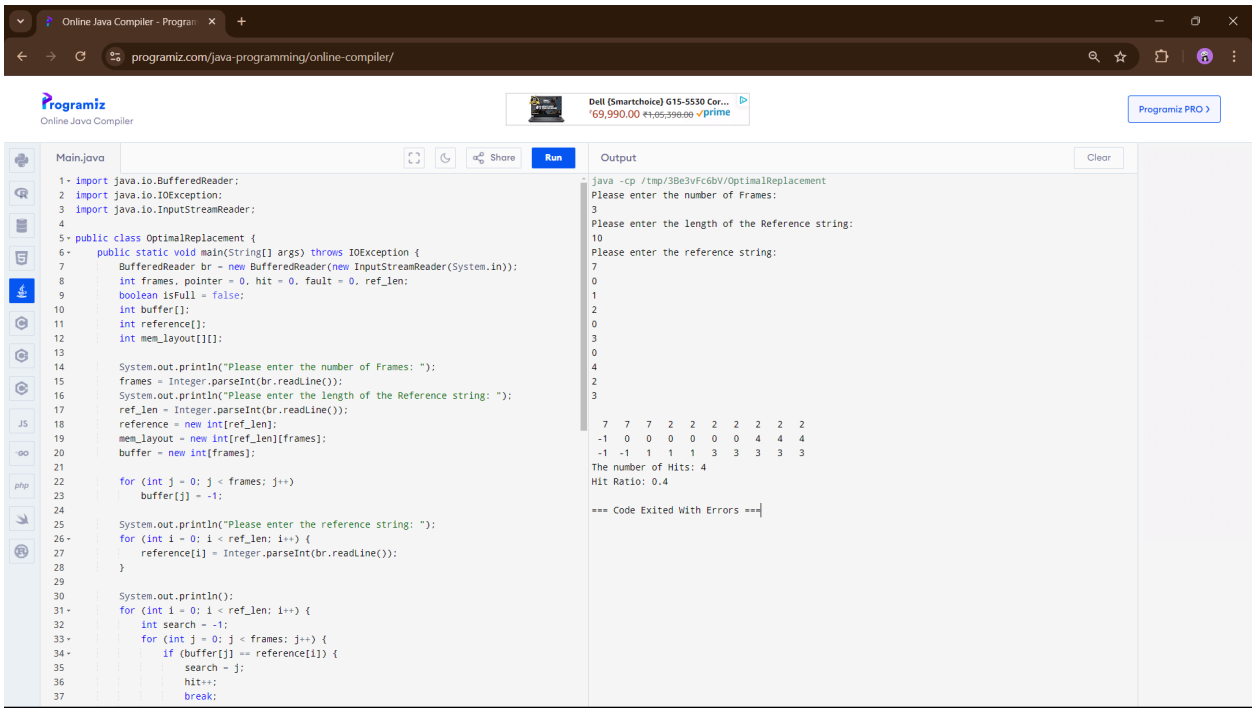
Code:-

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

public class OptimalReplacement {
   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;
      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];
}
for (int i = 0; i < frames; i++) {
    for (int j = 0; j < ref_len; j++)
        System.out.printf("%3d ", 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 :-


Online Java Compiler - Program
programiz.com/java-programming/online-compiler/
Programiz
Online Java Compiler
Dell (Smartchoice) G15-5530 Cor...
₹69,990.00 ₹1,05,390.00 prime
Programiz PRO
Main.java
Share
Run
Output
Clear
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class OptimalReplacement {
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;
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;
java -cp /tmp/3Be3vFc6bV/OptimalReplacement
Please enter the number of Frames:
3
Please enter the length of the Reference string:
10
Please enter the reference string:
7
0
1
2
0
3
0
4
2
3
7 7 7 2 2 2 2 2 2 2
-1 0 0 0 0 0 0 4 4 4
-1 -1 1 1 1 3 3 3 3 3
The number of Hits: 4
Hit Ratio: 0.4
=== Code Exited With Errors ===