

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++;
}

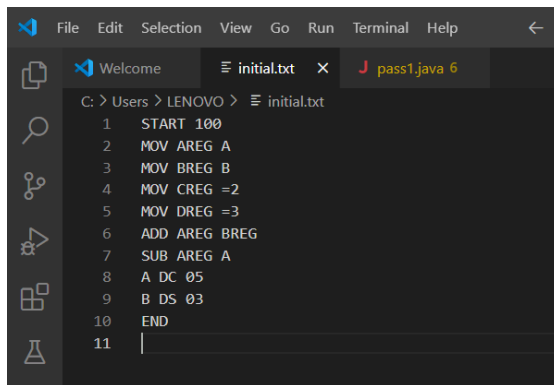
for (int i = 0; i < scount; i++) {
    p1.println(i + "\t" + sv[i] + "\t" + sadd[i]);
}
p1.close();

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

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

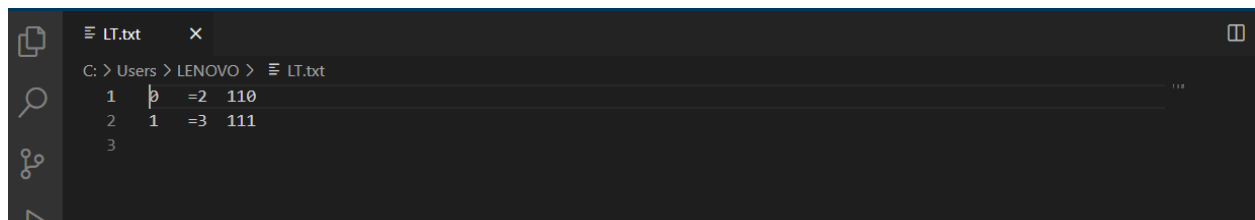
```

Initial :-

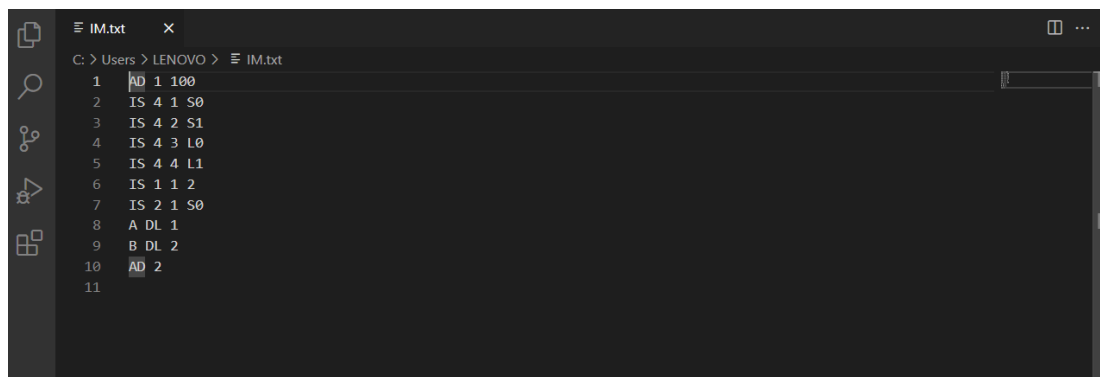


```
File Edit Selection View Go Run Terminal Help
Welcome initial.txt pass1.java 6
C: > Users > LENOVO > initial.txt
1 START 100
2 MOV AREG A
3 MOV BREG B
4 MOV CREG =2
5 MOV DREG =3
6 ADD AREG BREG
7 SUB AREG A
8 A DC 05
9 B DS 03
10 END
11
```

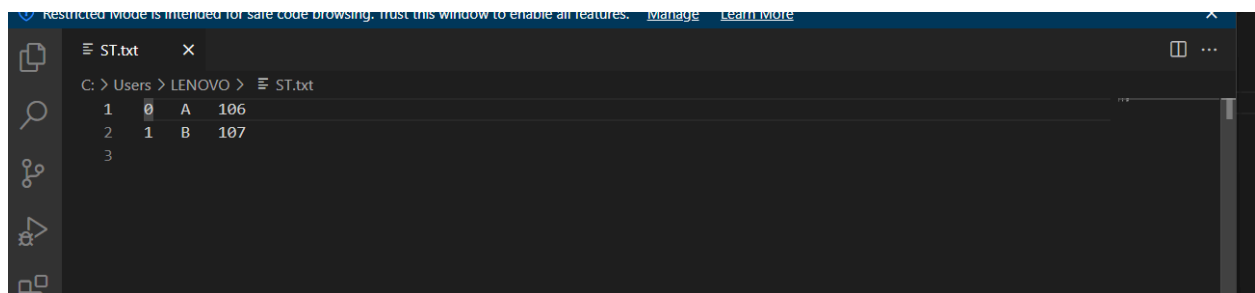
OUTPUT :-



```
LT.txt
C: > Users > LENOVO > LT.txt
1 0 =2 110
2 1 =3 111
3
```



```
IM.txt
C: > Users > LENOVO > IM.txt
1 AD 1 100
2 IS 4 1 S0
3 IS 4 2 S1
4 IS 4 3 L0
5 IS 4 4 L1
6 IS 1 1 2
7 IS 2 1 S0
8 A DL 1
9 B DL 2
10 AD 2
11
```



```
Restricted mode is intended for safe code browsing. Visit this window to enable all features. Manage Learn more
ST.txt
C: > Users > LENOVO > ST.txt
1 0 A 106
2 1 B 107
3
```

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
19
```

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\t\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 :-

The screenshot shows a web browser with the URL `programiz.com/java-programming/online-compiler/`. The page title is "Programiz Online Java Compiler". The code editor on the left contains the following Java code:

```

Main.java
34     finish[i] = sum;
35 }
36
37 // Calculate turnaround time and waiting time for each
   process
38 for (int i = 0; i < n; i++) {
39     turntt[i] = finish[i] - arrival[i];
40     total_tt += turntt[i];
41
42     wait[i] = turntt[i] - cpu[i];
43     total_waiting += wait[i];
44 }
45
46 // Display process details
47 System.out.println("\n\nProcess\t\tAT\t\tCPU_T");
48 for (int i = 0; i < n; i++) {
49     System.out.println(process[i] + "\t\t" + arrival[i] +
   "\t" + cpu[i]);
50 }
51
52 System.out.println("\n\n");
53 System.out.println("Average Turnaround Time: " + (total_tt /
   n));
54 System.out.println("Average Waiting Time: " + (total_waiting
   / n));
55 }
56 }
57

```

The output window on the right shows the following text:

```

java -cp /tmp/V2Exu0NxpV/fcfs
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
1	0	4
2	1	3
3	2	1

```

Average Turnaround Time: 5.3333335
Average Waiting Time: 2.6666667
=== Code Execution Successful ===

```

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 :-

The screenshot displays the Programiz Online Java Compiler interface. The left sidebar contains icons for various programming languages: Java, Python, JavaScript, C++, C, PHP, and others. The main editor area shows a Java file named 'Main.java' with the following code:

```
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\t\tCPU_T");
70 for (int i = 0; i < n; i++) {
71     System.out.println(process[i] + "\t\t" + arrival[i] +
72         "\t\t" + cpu[i]);
73 }
74 System.out.println("\n\n");
75 System.out.println("Average Turnaround Time: " + (total_tt /
76     n));
77 System.out.println("Average Waiting Time: " + (total_waiting
78     / n));
79 }
```

The right panel shows the 'Output' window with the following text:

```
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 ===

On the right side of the interface, there is a promotional banner for Dell (Smartchoice) laptops, featuring a 34% off discount and a 'Shop now' button.

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\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 :-

The screenshot displays a web-based Java IDE with the following components:

- Editor:** Contains the Java code for process scheduling, including arrival times, CPU times, and calculations for turnaround and waiting times.
- Output:** Shows the program's execution results, including the process execution sequence and the final average turnaround and waiting times.

Code Snippet (Main.java):

```

54         ncpu[i] = difference;
55     }
56
57     // Update the total execution time
58     if (ncpu[i] < t_quantum) {
59         sum += difference;
60     } else {
61         sum += t_quantum;
62     }
63
64     finish[k] = sum;
65     seq[k] = i;
66     System.out.print((seq[k] + 1) + " ");
67     k++;
68 }
69 }
70
71 System.out.println();
72
73 // Calculate turnaround time and waiting time for each process
74 for (int i = 0; i < n; i++) {
75     int carr = 0, tt = 0;
76     carr = arrival[i];
77
78     for (int j = 0; j < k; j++) {
79         if (seq[j] == i) {
80             tt += (finish[j] - carr);
81             carr = finish[j];
82         }
83     }
84
85     turntt[i] = tt;
86
87     System.out.println("Turn around time for Process " + (i + 1) + ": " +
88         turntt[i]);
89     total_tt += turntt[i];

```

Output:

```

- java -cp /tmp/QWUmqb419/robbin
Enter Number Of Processes You Want To Execute:
3
Enter arrival time of Process 1:
0
Enter CPU time of Process 1:
5
Enter arrival time of Process 2:
1
Enter CPU time of Process 2:
4
Enter arrival time of Process 3:
2
Enter CPU time of Process 3:
3
Enter time quantum:
2
Process execution sequence:
1 2 3 1 2 3 1

Turn around time for Process 1: 11
Waiting time for Process 1: 6
Turn around time for Process 2: 10
Waiting time for Process 2: 6
Turn around time for Process 3: 8
Waiting time for Process 3: 5

Process      AT      CPU_T
1            0        5
2            1        4
3            2        3

Average Turnaround Time: 9.67
Average Waiting Time: 5.67

```

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\t\tCPU_T\t\tPriority");
for (int i = 0; i < n; i++) {
    System.out.println(process[i] + "\t\t" + arrival[i] + "\t\t" + cpu[i] + "\t\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 :-

The screenshot shows an online Java compiler interface. The code in the editor implements a priority scheduling algorithm for three processes. The output shows the execution flow where Process 2 is executed first due to its highest priority (1), followed by Process 1 (priority 2), and then Process 3 (priority 3). The final output includes the average turnaround and waiting times for all processes.

```
45 process[j] = temp;
46
47     temp = pri[i];
48     pri[i] = pri[j];
49     pri[j] = temp;
50 }
51 }
52
53 // Calculating finish time for each process
54 for (int i = 0; i < n; i++) {
55     sum += cpu[i];
56     finish[i] = sum;
57 }
58
59 // Calculating 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 information
69 System.out.println("\n\nProcess\t\tAT\tCPU_T\tPriority");
70 for (int i = 0; i < n; i++) {
71     System.out.println(process[i] + "\t\t" + arrival[i] + "\t" + cpu[i] + "\t" +
72         pri[i]);
73 }
74
75 System.out.println("\n\n");
76 System.out.println("Average Turnaround Time: " + (total_tt / n));
77 System.out.println("Average Waiting Time: " + (total_waiting / n));
78 }
79 }
80
```

Output

```
java -cp /tmp/1YbCp80U1Z/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:
3
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: 7.3333333
Average Waiting Time: 3.3333333
=== Code Execution Successful ===
```

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 :-

The screenshot shows an online Java compiler interface with the following components:

- Header:** Includes the Programiz logo, a banner for "VICTUS-HP SPECIAL EDITION WITH RTX 3050", and a "Programiz PRO" button.
- Editor:** Contains the Java code for the FIFO algorithm, with line numbers 31 to 67. The code processes a reference string and calculates hits, faults, and hit ratio.
- Output Console:** Displays the execution results, including prompts for the number of frames, reference string length, and reference string, followed by the memory layout and final statistics.
- Footer:** Includes a sidebar with various development tools and a banner for "VICTUS-HP SPECIAL EDITION WITH RTX 3050".

Output Console Content:

```

java -cp /tmp/SgnUAfYc4/FIFO
Please enter the number of Frames:
3
Please enter the length of the Reference string:
7
Please enter the reference string:
1
3
0
3
5
6
3

The Memory Layout is:
 1  1  1  1  5  5  5
-1  3  3  3  3  6  6
-1 -1  0  0  0  0  3

The number of Hits: 1
Hit Ratio: 0.14285715
The number of Faults: 6

--- Code Execution Successful ---

```

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 :-

The screenshot shows an online Java compiler interface. The code in Main.java implements an LRU (Least Recently Used) page replacement algorithm. The output window shows the following execution details:

```

java -cp /tmp/HT0LM3B00C/LruAlgo
Enter page table size:
6
Enter elements in page table:
1 3 0 3 5 6
Enter page frame size:
3
Page: 1
[1] [ ] [ ]
Page: 3
[1] [3] [ ]
Page: 0
[1] [3] [0]
Page: 3
[1] [3] [0]
Page: 5
[5] [3] [0]
Page: 6
[5] [3] [6]

Number of page faults: 2

[5] [3] [6]

=== Code Execution Successful ===

```

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) {
```



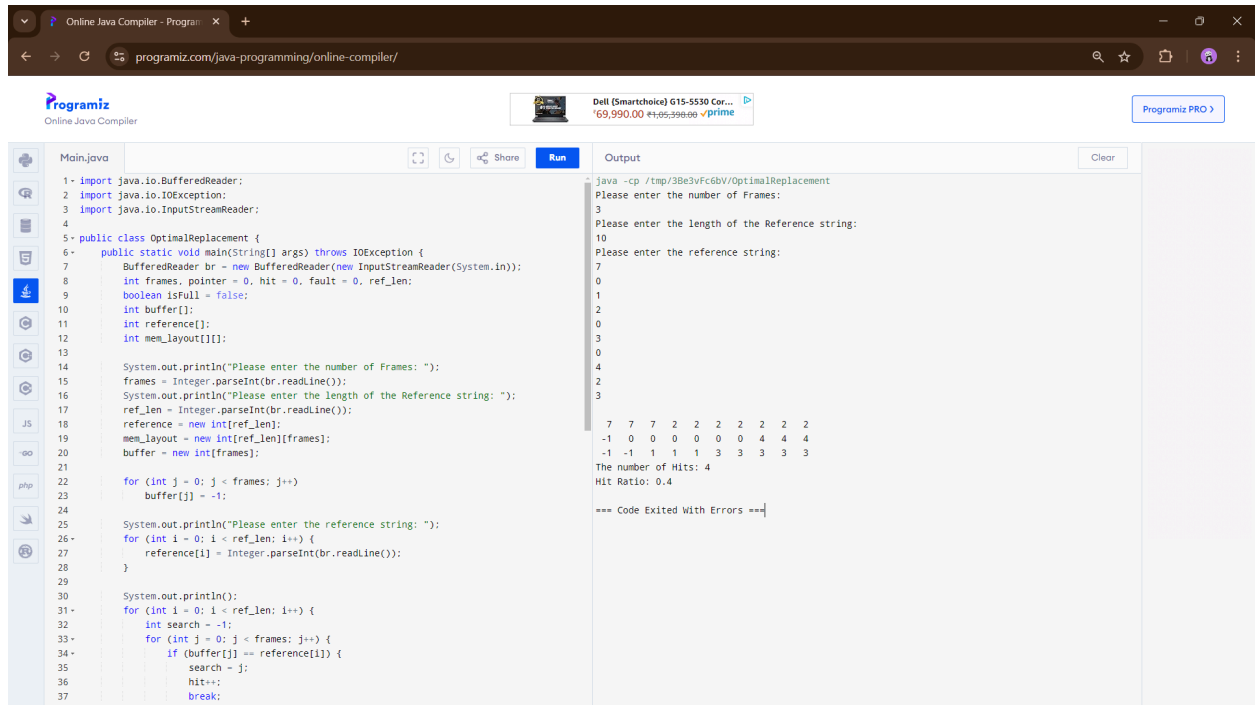
```

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 :-



The screenshot shows a web browser window with the URL `programiz.com/java-programming/online-compiler/`. The page features the Programiz logo and a navigation bar. The main content area is divided into two panels: a code editor on the left and an output window on the right.

The code editor displays the following Java code:

```
1- import java.io.BufferedReader;  
2- import java.io.IOException;  
3- import java.io.InputStreamReader;  
4  
5- public class OptimalReplacement {  
6-     public static void main(String[] args) throws IOException {  
7         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
8         int frames, pointer = 0, hit = 0, fault = 0, ref_len;  
9         boolean isFull = false;  
10        int buffer[];  
11        int reference[];  
12        int mem_layout[];  
13  
14        System.out.println("Please enter the number of Frames: ");  
15        frames = Integer.parseInt(br.readLine());  
16        System.out.println("Please enter the length of the Reference string: ");  
17        ref_len = Integer.parseInt(br.readLine());  
18        reference = new int[ref_len];  
19        mem_layout = new int[ref_len][frames];  
20        buffer = new int[frames];  
21  
22        for (int j = 0; j < frames; j++)  
23            buffer[j] = -1;  
24  
25        System.out.println("Please enter the reference string: ");  
26        for (int i = 0; i < ref_len; i++) {  
27            reference[i] = Integer.parseInt(br.readLine());  
28        }  
29  
30        System.out.println();  
31        for (int i = 0; i < ref_len; i++) {  
32            int search = -1;  
33            for (int j = 0; j < frames; j++) {  
34                if (buffer[j] == reference[i]) {  
35                    search = j;  
36                    hit++;  
37                    break;  
38                }  
39            }  
40            if (search == -1) {  
41                fault++;  
42                mem_layout[i][frames - 1] = reference[i];  
43                for (int k = 0; k < frames - 1; k++)  
44                    mem_layout[i][k] = mem_layout[i][k + 1];  
45                buffer[frames - 1] = reference[i];  
46            }  
47        }  
48        System.out.println("The number of Hits: " + hit);  
49        System.out.println("Hit Ratio: " + (hit / ref_len));  
50    }  
51 }
```

The output window displays the following text:

```
java -cp /tmp/38e3vFc6bV/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 ===
```