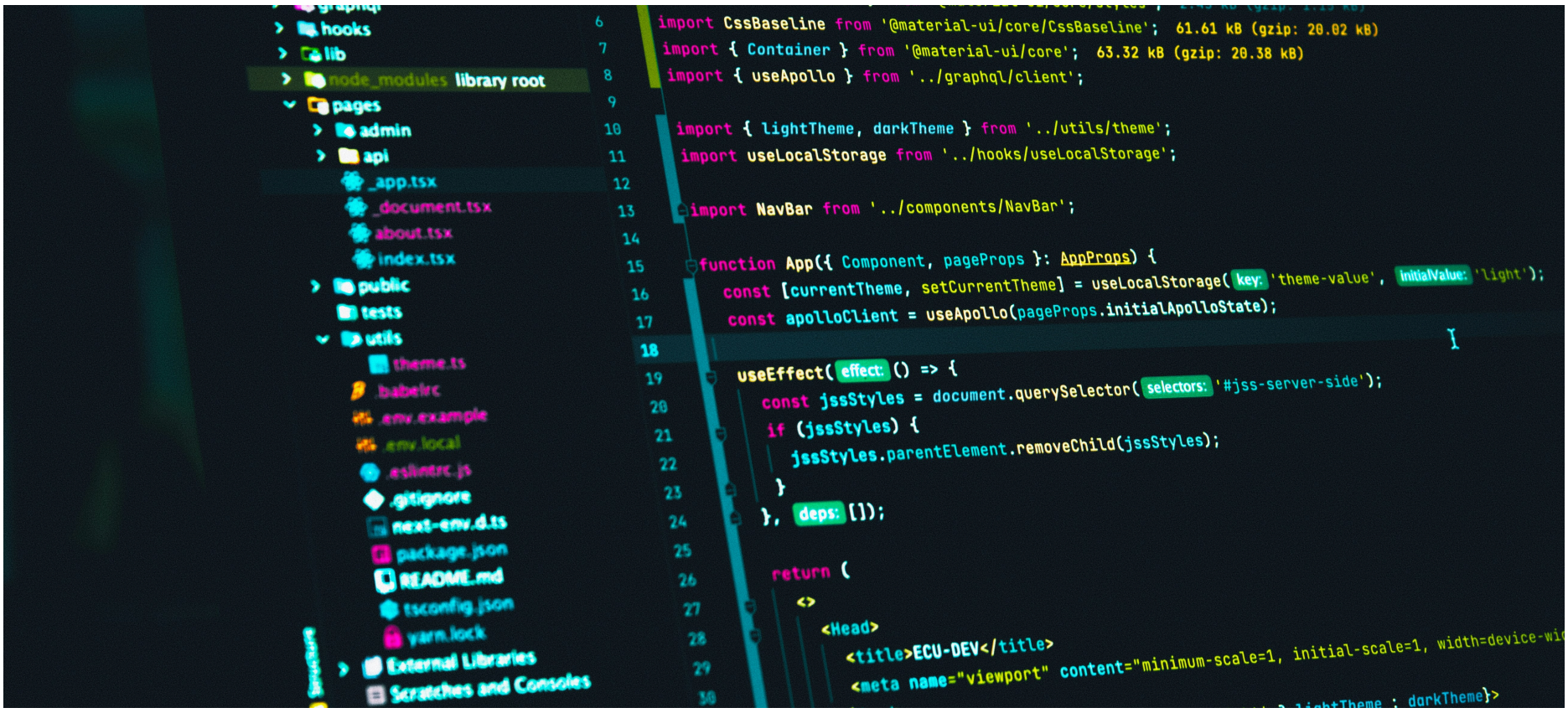


Phase 1

Source Code

```
import java.io.*;
import java.util.*;
class phase1{
	public static char[][] memory=new char[100][4];
	public static int IC;
	public static char[] R=new char[4];
	public static char[] IR=new char[4];
	public static int SI;
	public static int C;
	public static int numberOfInstructions;
	public static void main(String[] args) throws Exception{
		int i;
		load();
	}

	public static void displayMemoryContents(){
		int i;
		for(i=0; i<100; i++){
			if(i%10==0 && i!=0)
				System.out.println("------------------------------------------------------------------");
			System.out.println(i+"  "+memory[i][0]+memory[i][1]+memory[i][2]+memory[i][3]);
		}
		System.out.println();
		System.out.println();
		System.out.println();
	}

	public static void load() throws Exception{
		String line;
		int i;
		String subline;
		int linePointer=0;
		int insCounter=0;
		File f=new File("input.txt");
```

```
            Scanner sc=new Scanner(f);
            while(sc.hasNextLine()){
                  line=sc.nextLine();
                  if(line.length()>4)
                        subline=line.substring(0,4);
                  else
                        subline=line;
                  if(subline.equals("$AMJ")){
                        linePointer=0;
                        insCounter=0;
                        numberOfInstructions=Integer.parseInt(line.substring(8,12));
                        initialize();
                  }
                  else if(subline.equals("$DTA")){
                        executeProgram(sc);
                  }
                  else if(subline.equals("$END")){
                        displayMemoryContents();
                        continue;
                  }
                  else{
                        line=line.trim();
                        char[] instructions=line.toCharArray();
                        for(i=0; i<instructions.length; i++){
                              if(insCounter!=0 && insCounter%4==0)
                                    linePointer++;
                              if(instructions[i]=='H'){
                                    memory[linePointer][insCounter%4]=instructions[i];
                                    insCounter=((insCounter/4)+1)*4;
                              }
                              else{
                                    memory[linePointer][insCounter%4]=instructions[i];
                                    insCounter++;
                              }
                        }
                  }
            }
      }

      public static void initialize() throws Exception{
            int i;
            IC=0;
            SI=0;
            C=0;
            R[0]='_';
            R[1]='_';
            R[2]='_';
            R[3]='_';
            IR[0]='_';
            IR[1]='_';
            IR[2]='_';
            IR[3]='_';
            for(i=0; i<100; i++){
                  memory[i][0]='_';
                  memory[i][1]='_';
                  memory[i][2]='_';
                  memory[i][3]='_';
            }
```

```
        File of=new File("output.txt");
        if(of.createNewFile())
            System.out.println("Created an output file.");
    }

    public static void executeProgram(Scanner sc) throws Exception{
        int i,address;
        String temporary;
        while(IC!=numberOfInstructions){
            temporary="";
            address=0;
            for(i=0; i<4; i++)
                IR[i]=memory[IC][i];
            if(IR[0]=='G' && IR[1]=='D'){
                IC++;
                IR[3]='0';
                SI=1;
                masterMode(sc);
            }
            else if(IR[0]=='P' && IR[1]=='D'){
                IC++;
                IR[3]='0';
                SI=2;
                masterMode(sc);
            }
            else if(IR[0]=='H'){
                IC++;
                SI=3;
                masterMode(sc);
                break;
            }
            else if(IR[0]=='L' && IR[1]=='R'){
                IC++;
                temporary=""+IR[2]+IR[3];
                address=Integer.parseInt(temporary);
                for(i=0; i<4; i++)
                    R[i]=memory[address][i];
            }
            else if(IR[0]=='S' && IR[1]=='R'){
                IC++;
                temporary=""+IR[2]+IR[3];
                address=Integer.parseInt(temporary);
                for(i=0; i<4; i++)
                    memory[address][i]=R[i];
            }
            else if(IR[0]=='C' && IR[1]=='R'){
                IC++;
                temporary=""+IR[2]+IR[3];
                address=Integer.parseInt(temporary);
                for(i=0; i<4; i++){
                    if(R[i]==memory[address][i]){
                        C=1;
                    }
                    else{
                        C=0;
                        break;
                    }
                }
```

```
            }
            else if(IR[0]=='B' && IR[1]=='T'){
                if(C==1){
                    temporary=""+IR[2]+IR[3];
                    address=Integer.parseInt(temporary);
                    IC=address;
                }
                else{
                    IC++;
                }
            }
        }
    }

    public static void masterMode(Scanner sc) throws Exception{
        if(SI==1)
            read(sc);
        else if(SI==2)
            write();
        else if(SI==3)
            terminate();
    }

    public static void read(Scanner sc) throws Exception{
        String data=sc.nextLine();
        char[] dataChar=data.toCharArray();
        int length=dataChar.length;
        int block=Integer.parseInt(""+IR[2]+IR[3]);
        int counterForArray=0;
        int counterForMemory=0;
        while(counterForArray!=length){
            if(dataChar[counterForArray]==' ' && counterForMemory!=0){
                block++;
                counterForArray++;
                counterForMemory=((counterForMemory/4)+1)*4;
            }
            else if(counterForMemory%4==0 && counterForMemory!=0){
                block++;
            }
            memory[block][counterForMemory%4]=dataChar[counterForArray];
            counterForArray++;
            counterForMemory++;
        }
    }

    public static void write() throws Exception{
        int block=Integer.parseInt(""+IR[2]+IR[3]);
        int i,j;
        String dataToWrite="";
        for(i=block; i<block+10; i++){
            for(j=0; j<4; j++){
                if(memory[i][j]!='_'){
                    dataToWrite=dataToWrite+memory[i][j];
                }
            }
            if(memory[i][j-1]=='_')
                dataToWrite=dataToWrite+" ";
        }
```

```
            FileWriter fw=new FileWriter("output.txt",true);
            dataToWrite=dataToWrite+"\n";
            fw.write(dataToWrite);
            fw.close();
        }

        public static void terminate() throws Exception{
            FileWriter fw=new FileWriter("output.txt",true);
            fw.write("\n\n");
            fw.close();
        }
}
```

All Five Jobs at once

Documentation

High Level Overview

1. We start reading the input file line by line
2. If the line contains “$AMJ” —> initialize the registers and memory
3. If the line contains “$DTA” —> start executing the program by reading instructions from the memory. (Instructions are already stored in the memory before we encounter $DTA).
4. If the line contains “$END” —> print two blank lines to the output file.
5. If it is none of the above, it is surely the program card.
6. We store those characters in the memory starting from location 0.
7. After these instructions, we will surely encounter $DTA.
8. When we encounter $DTA, we start the `executeProgram()` function. What we do in this function is read the instructions from the memory into the IR register, then check what instruction is it.
9. If it is GD, PD, or H, we call the `masterMode()` function, as these instructions cannot be executed normally. Suitable System Interrupt values are given to each of these instructions.
10. If it is BT, CR, SR, or LR, respective tasks are performed. At the end the output is written to the output.txt file.

Description of each function

`load()`

- Reads the input file line by line

- Calls `initialize()` upon encountering **$AMJ,** as it signals the beginning of a new job.
- **Loads the instructions from the input file to the main memory.**
- Calls `executeProgram()` upon encountering **$DTA,** as it means that all the instructions that were present between $AMJ and $DTA are stored in the main memory.
- Calls `displayMemoryContents()` upon encountering **$END**.

initialize()

- Initializes SI, IC to 0
- Initializes the whole memory array, and registers to `_` (underscore) to depict empty memory space.

executeProgram()

- Checks the value of IC, loads the instruction present at `[IC]` into the IR.
- Calls `masterMode()` after setting **SI=1**, if it is **GD**.
- Calls `masterMode()` after setting **SI=2**, if it is **PD**.
- Calls `masterMode()` after setting **SI=3**, if it is **H**.
- Loads the R register with the content of memory location specified along with **LR**.
- Stores the value in R register to the memory location specifies along with **SR**.
- Compares the content present in the R register and at the memory location specified along with **CR**. Sets the toggle register if true.
- Updates the value of IC with the address given with **BT,** if the toggle register was set.
- Increments IC to fetch the next instruction.

masterMode()

- Calls `read()` if SI=1.
- Calls `write()` if SI=2.
- Calls `terminate()` if SI=3.

read()

- Reads the data card from the input file and stores it at the location specified with the instruction GD.

write()

- Writes the data from the memory location specified with the instruction PD, to the output file.

terminate()

- Prints two blank lines in the output file.

displayMemoryContents()

- Displays the content of our main memory, from line 0-99.

Key Problems

1. Storing the instructions in memory such that H is stored on one line and the next instruction gets stored on the next line instead of the same line

```
13      BT38
14      CR65
15      BT43
16      PD60
17      H___
18      LR80
19      SR60
-------------------------
20      CR80
21      LR70
22      BT06
```

2. While reading data from the data card, store the data after a space character on the new line in the memory.

```
$DTA
2023 was amazing
2023
2024
$END0002
```

```
-----------------------------------------------------------------------
60      2024
61      was_
62      amaz
63      ing_
64      ____
65      ____
66      ____
67      ____
68      ____
69      ____
-----------------------------------------------------------------------
70      2023
```

3. Making sure the data is written in a similar manner in the output card : If a word is on the new line, print it after a space character.

Solutions

1. Storing instructions correctly in the memory

```
public static void load() throws Exception{
        String line;
        int i;
        String subline;
        int linePointer=0;
        int insCounter=0;
        File f=new File("input.txt");
        Scanner sc=new Scanner(f);
        while(sc.hasNextLine()){
                line=sc.nextLine();
                if(line.length()>4)
                        subline=line.substring(0,4);
                else
                        subline=line;
                if(subline.equals("$AMJ")){
                        numberOfInstructions=Integer.parseInt(line.substring(8,12));
                        initialize();
                }
                else if(subline.equals("$DTA")){
                        executeProgram(sc);
                }
                else if(subline.equals("$END")){
                        break;
                }
                else{
                        line=line.trim();
                        char[] instructions=line.toCharArray();
                        for(i=0; i<instructions.length; i++){
                                if(insCounter!=0 && insCounter%4==0)
                                        linePointer++;
                                if(instructions[i]=='H'){
                                        memory[linePointer][insCounter%4]=instructions[i];
                                        insCounter=((insCounter/4)+1)*4;
                                }
                                else{
                                        memory[linePointer][insCounter%4]=instructions[i];
```

```
                        insCounter++;
                    }
                }
            }
        }
    }
```

We store a single line read from the file in the variable `line` (String). We extract the first 4 characters from this line and store it in the variable `subline` (String). We compare `subline` to $AMJ, $DTA, and $END. If it is equal to $AMJ, we initialize the memory with `_` , this is used to represent that the memory is empty. We initialize the toggle register to 0 (C=0), the Instruction Counter (IC) to 0 etc.

`linePointer` is used to store the line number in the memory where we have to store out instruction

After $AMJ are the instructions. We store these instructions in the memory starting from location 0. Each line in the memory has 4 bytes, and 10 lines form one block. Each instruction is of 4 bytes, except `HALT` .

Each letter (character) in “GD30” occupies one byte. G will occupy 1 byte, D will occupy 1 byte and so on.

Therefore, we need to make an exception when storing H in the memory, otherwise something like this might occur :

Let’s say our instructions were : GD10PD10HPD30. This is how ideally it should have been stored in the memory :

GD10

PD10

H

PD30

But if we do not make an exception while storing `H` , it will be stored like :

GD10

PD10

HPD3

0

2. Writing data from data card to the memory (similar to the above one)

```
public static void read(Scanner sc) throws Exception{
        String data=sc.nextLine();
        char[] dataChar=data.toCharArray();
        int length=dataChar.length;
        int block=Integer.parseInt(""+IR[2]+IR[3]);
        int counterForArray=0;
        int counterForMemory=0;
        while(counterForArray!=length){
            if(dataChar[counterForArray]==' ' && counterForMemory!=0){
                block++;
                counterForArray++;
                counterForMemory=((counterForMemory/4)+1)*4;
            }
            else if(counterForMemory%4==0 && counterForMemory!=0){
                block++;
            }
            memory[block][counterForMemory%4]=dataChar[counterForArray];
            counterForArray++;
            counterForMemory++;
         }
}
```

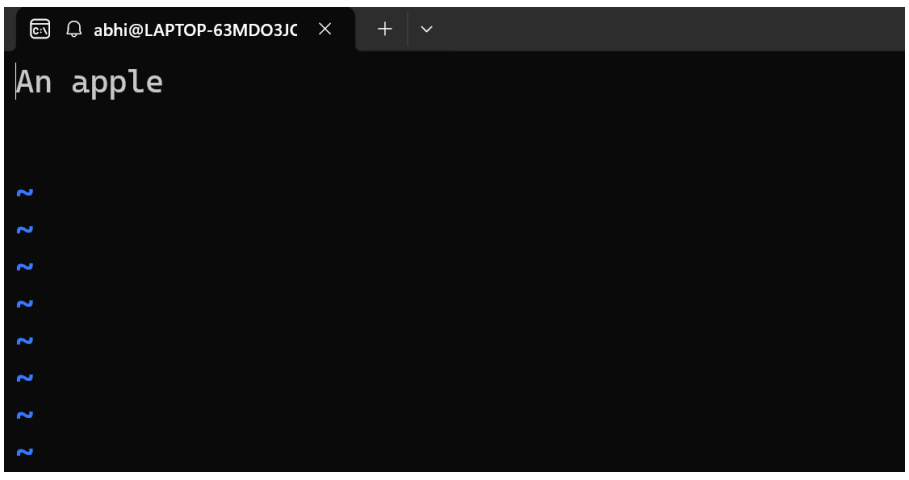
3. Writing data to the output file

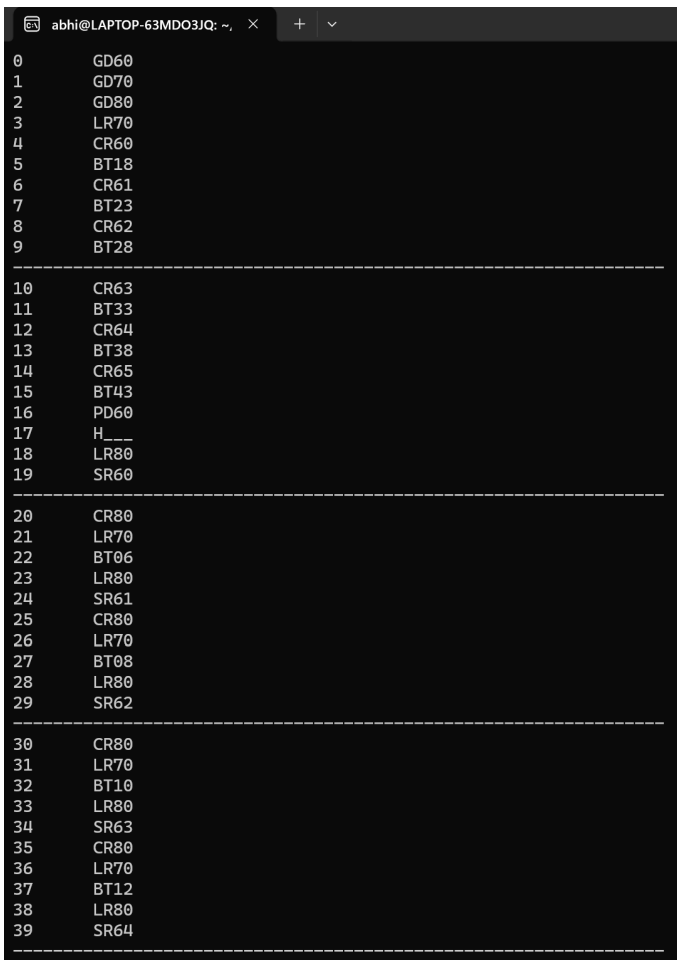
```
public static void write() throws Exception{
        int block=Integer.parseInt(""+IR[2]+IR[3]);
        int i,j;
        String dataToWrite="";
        for(i=block; i<block+10; i++){
                for(j=0; j<4; j++){
                        if(memory[i][j]!='_'){
                                dataToWrite=dataToWrite+memory[i][j];
                        }
                }
                if(memory[i][j-1]=='_')
                        dataToWrite=dataToWrite+" ";
        }
        FileWriter fw=new FileWriter("output.txt",true);
        dataToWrite=dataToWrite+"\n";
        fw.write(dataToWrite);
        fw.close();
}
```

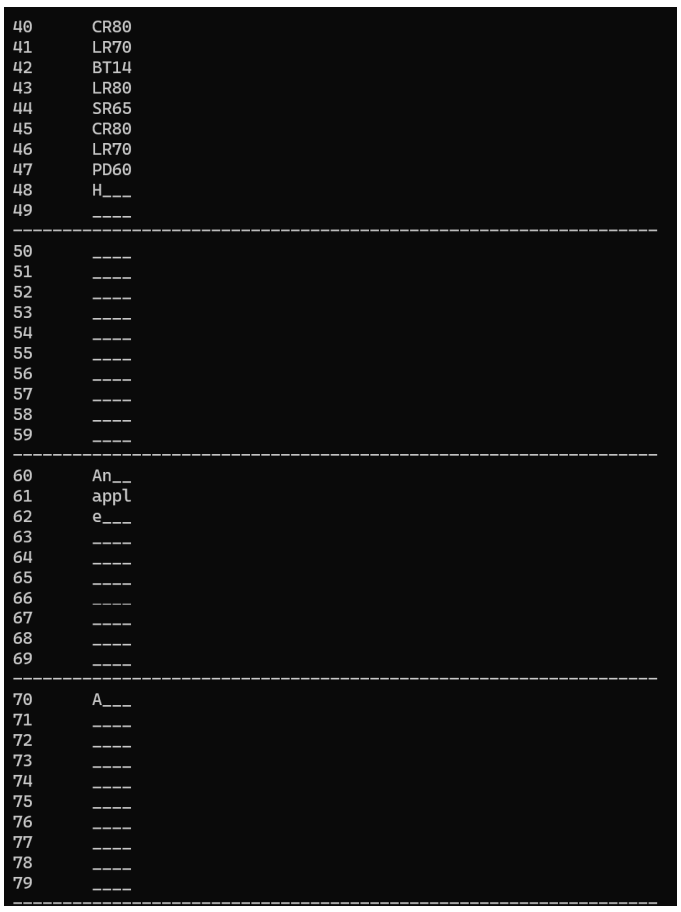
Testing our Jobs

Replacing one word with another given word

$AMJ000200480002
GD60GD70GD80LR70CR60BT18CR61BT23CR62BT28
CR63BT33CR64BT38CR65BT43PD60HLR80SR60
CR80LR70BT06LR80SR61CR80LR70BT08LR80SR62
CR80LR70BT10LR80SR63CR80LR70BT12LR80SR64
CR80LR70BT14LR80SR65CR80LR70PD60H
$DTA
A apple
A
An
$END0002

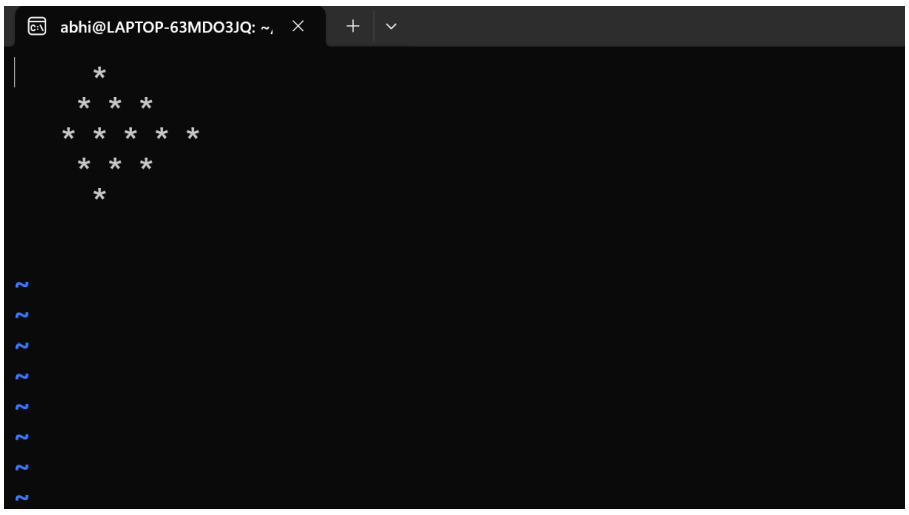


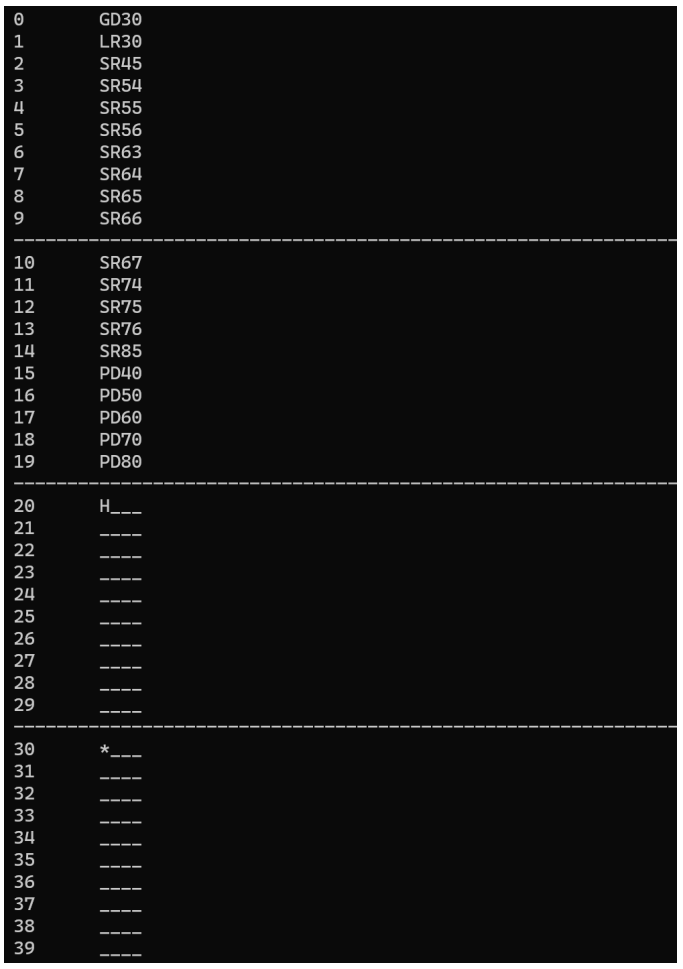


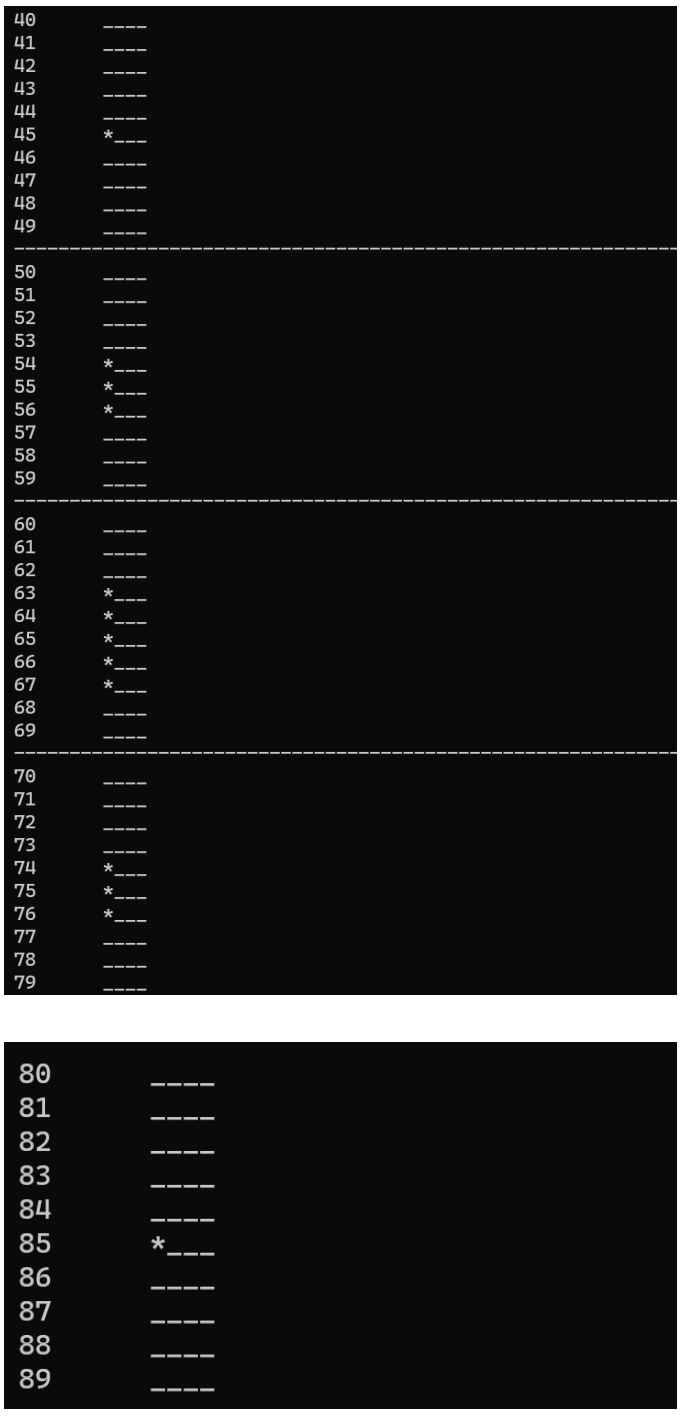


Diamond Pattern

$AMJ000300210005
GD30LR30SR45SR54SR55SR56SR63SR64SR65SR66
SR67SR74SR75SR76SR85PD40PD50PD60PD70PD80
H
$DTA
*
$END0003







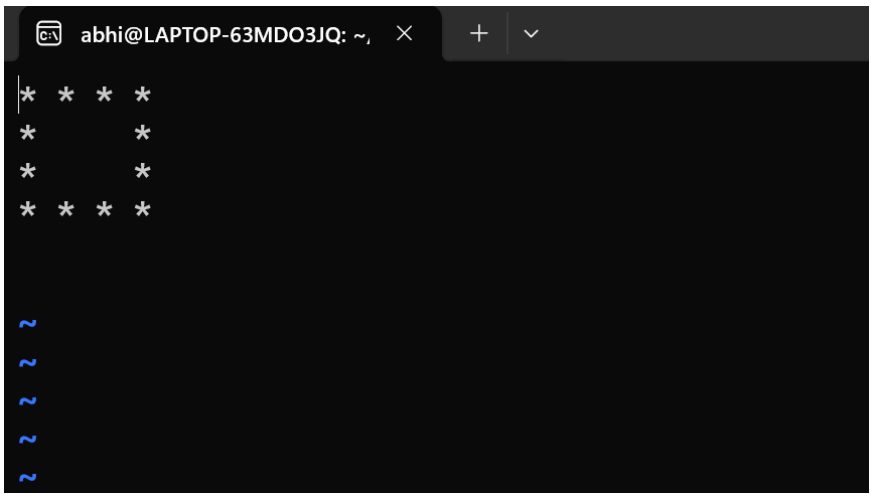
Hollow Square

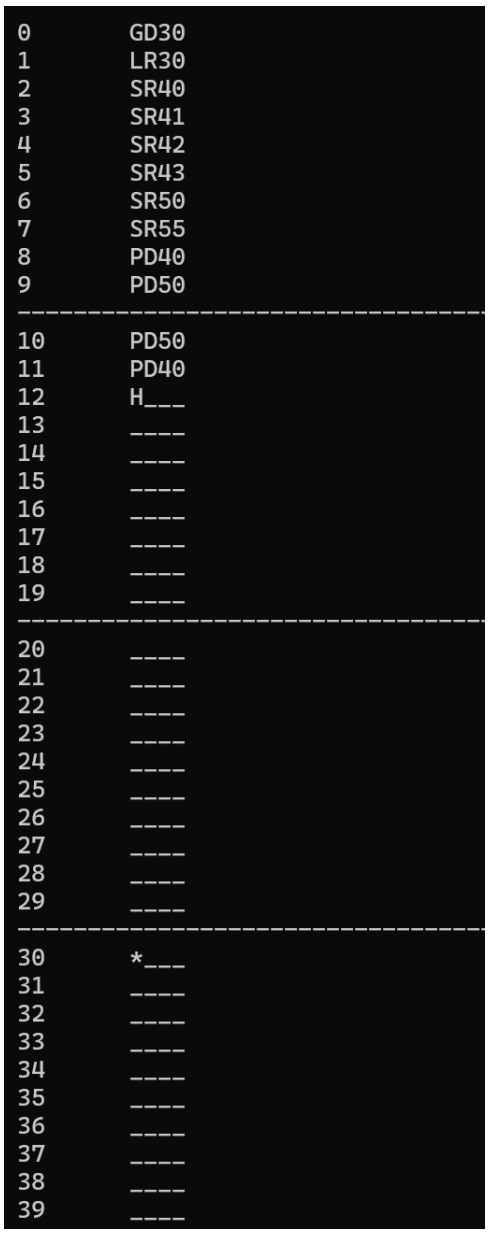
$AMJ000400130004

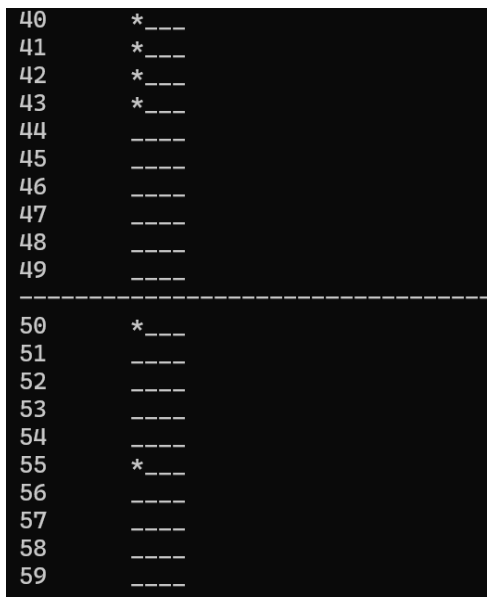
GD30LR30SR40SR41SR42SR43SR50SR55PD40PD50

PD50PD40H

$DTA
*
$END0004



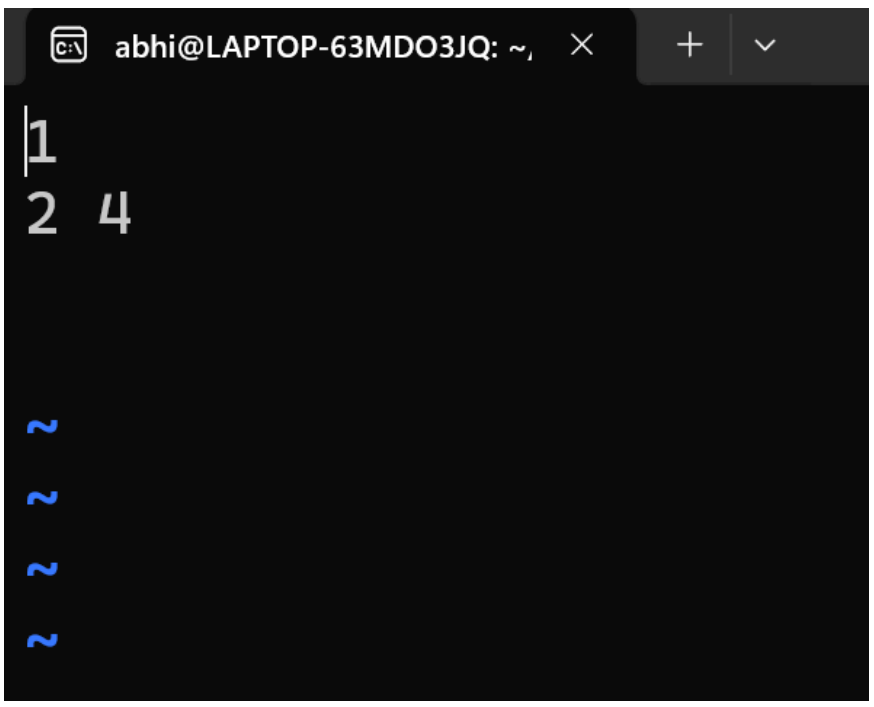




Print all non zero elements of the matrix

$AMJ000500260002
GD30GD40GD50LR30CR40BT09LR40SR60LR30CR41
BT14LR41SR61LR30CR50BT19LR50SR70LR30CR51
BT23LR51SR71PD60PD70H
$DTA
0
1 0
2 4

$END0005



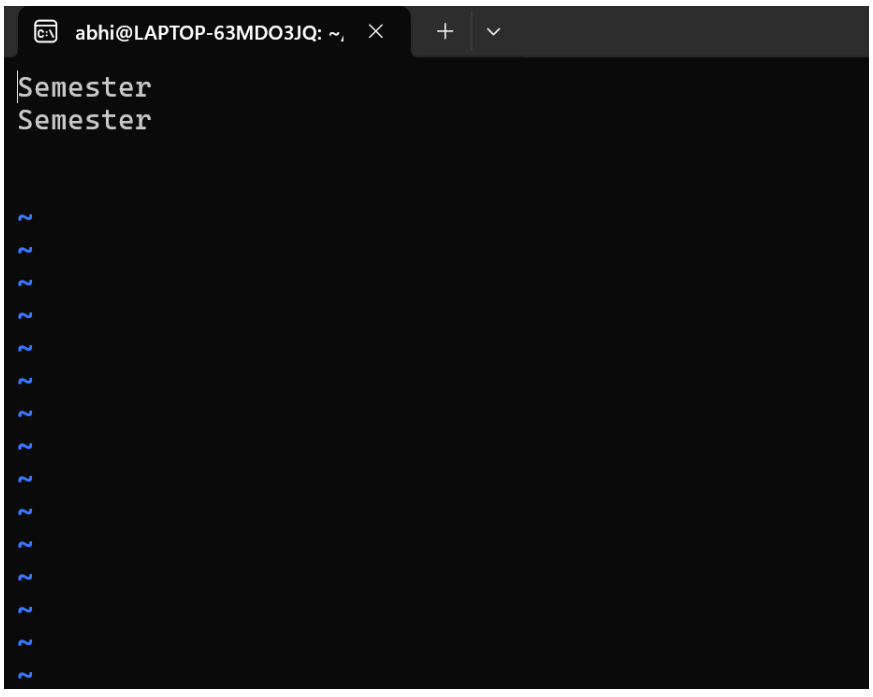

```
0       GD30
1       GD40
2       GD50
3       LR30
4       CR40
5       BT09
6       LR40
7       SR60
8       LR30
9       CR41
-----------------------------------------
10      BT14
11      LR41
12      SR61
13      LR30
14      CR50
15      BT19
16      LR50
17      SR70
18      LR30
19      CR51
-----------------------------------------
20      BT23
21      LR51
22      SR71
23      PD60
24      PD70
25      H___
26      ____
27      ____
28      ____
29      ____
30      0___
31      ____
32      ____
33      ____
34      ____
35      ____
36      ____
37      ____
38      ____
39      ____
-----------------------------------------
40      1___
41      0___
42      ____
43      ____
44      ____
45      ____
46      ____
47      ____
48      ____
49      ____
-----------------------------------------
50      2___
51      4___
52      ____
53      ____
54      ____
55      ____
56      ____
57      ____
58      ____
59      ____
```

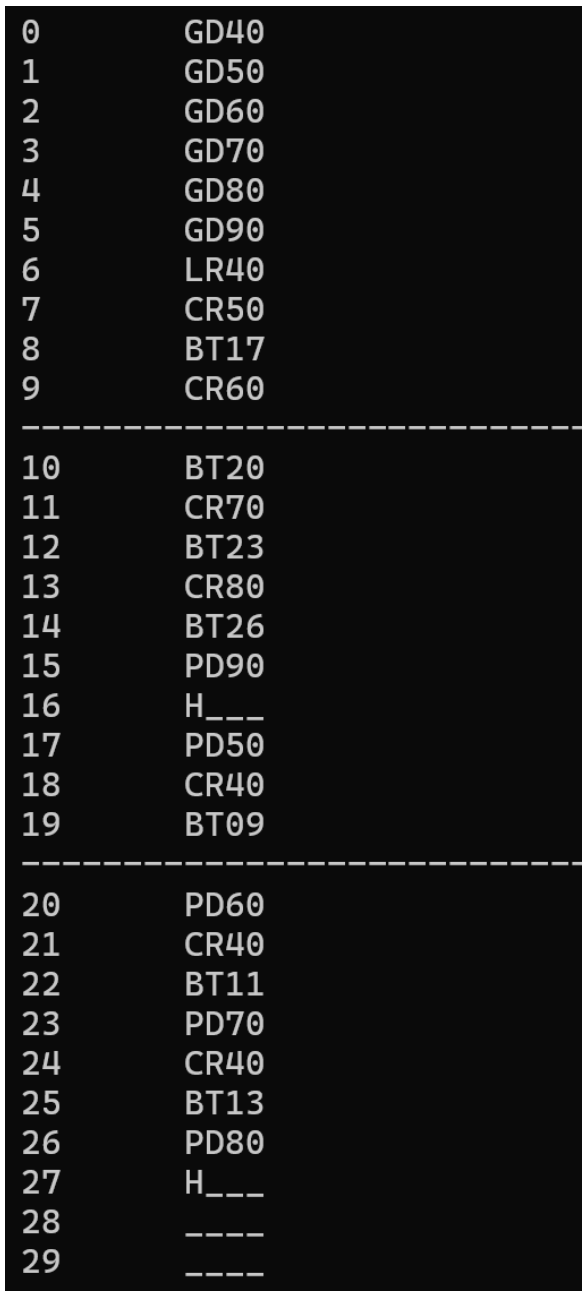
```
60      1___
61      ____
62      ____
63      ____
64      ____
65      ____
66      ____
67      ____
68      ____
69      ____
-----------------------------------
70      2___
71      4___
72      ____
73      ____
74      ____
75      ____
76      ____
77      ____
78      ____
79      ____
```

All occurrences of a particular word in the data

$AMJ000100280005
GD40GD50GD60GD70GD80GD90LR40CR50BT17CR60
BT20CR70BT23CR80BT26PD90HPD50CR40BT09
PD60CR40BT11PD70CR40BT13PD80H
$DTA
Semester

Word1
Word2
Semester
Semester
No Match Found
$END0001





```
40      Seme
41      ster
42      ____
43      ____
44      ____
45      ____
46      ____
47      ____
48      ____
49      ____
--------------------------------
50      Word
51      1___
52      ____
53      ____
54      ____
55      ____
56      ____
57      ____
58      ____
59      ____
--------------------------------
60      Word
61      2___
62      ____
63      ____
64      ____
65      ____
66      ____
67      ____
68      ____
69      ____
--------------------------------
70      Seme
71      ster
72      ____
73      ____
74      ____
75      ____
76      ____
77      ____
78      ____
79      ____
```

```
80      Seme
81      ster
82      ____
83      ____
84      ____
85      ____
86      ____
87      ____
88      ____
89      ____
----------------------------
90      No__
91      Matc
92      h___
93      Foun
94      d___
95      ____
96      ____
97      ____
98      ____
99      ____
```