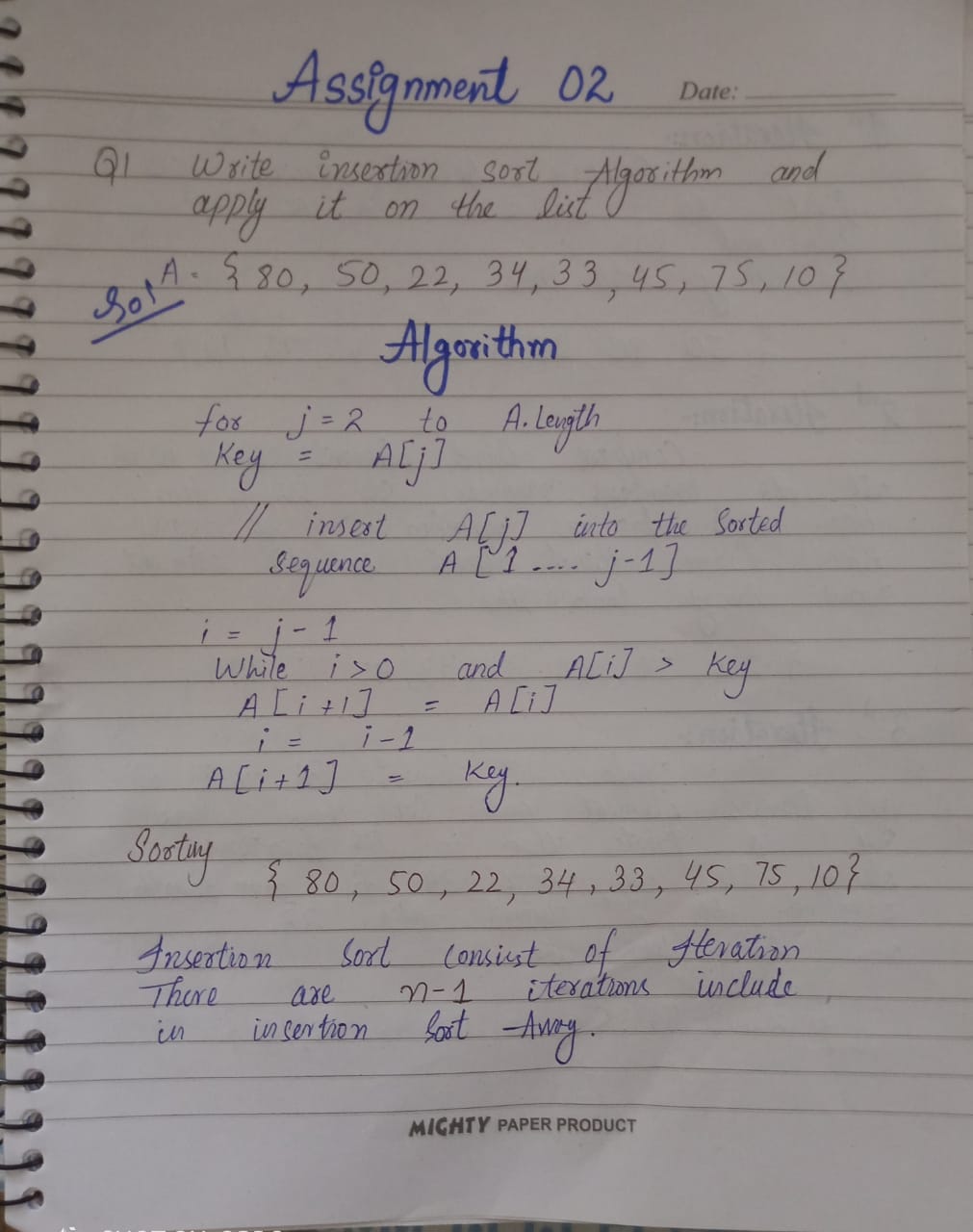
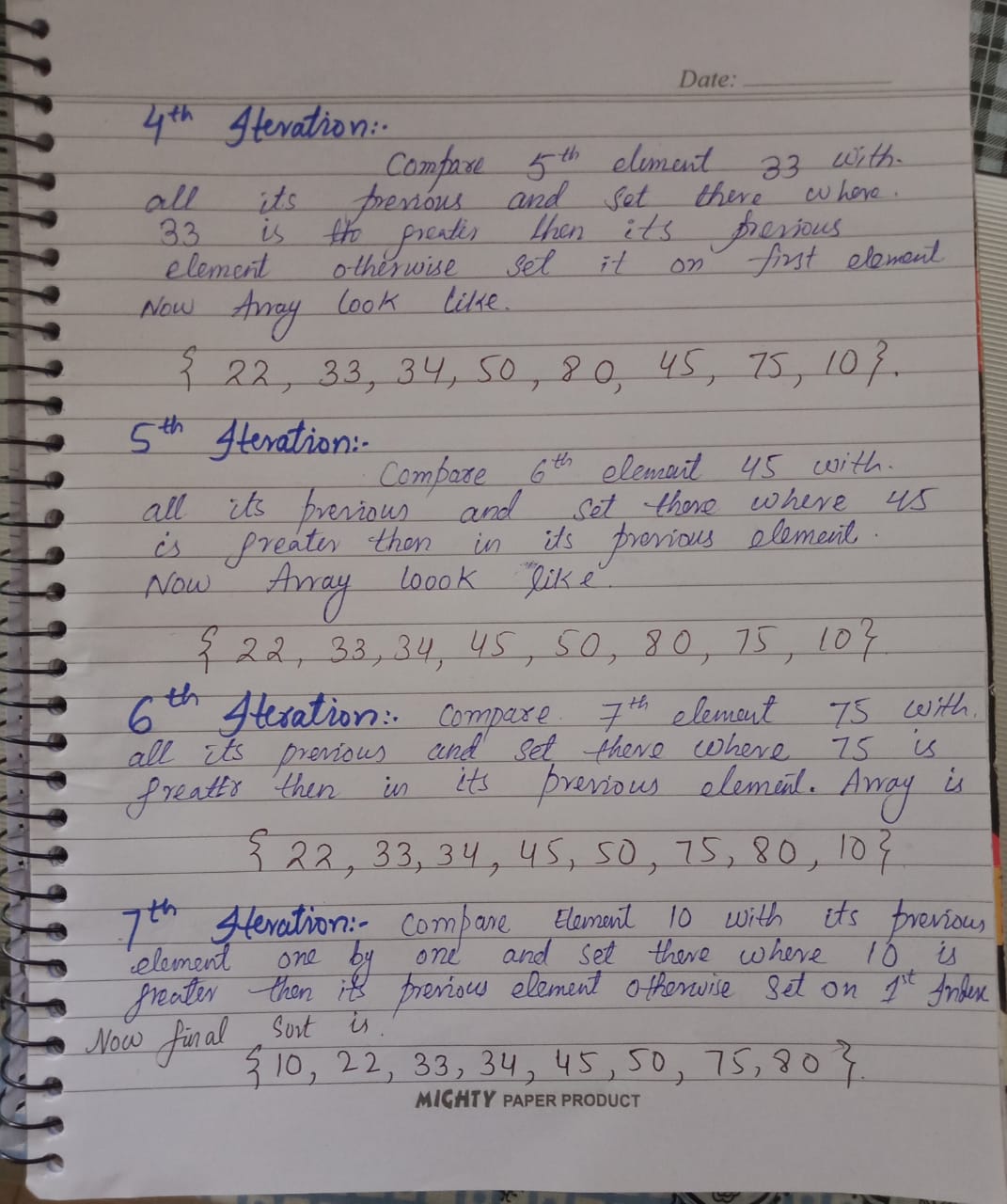
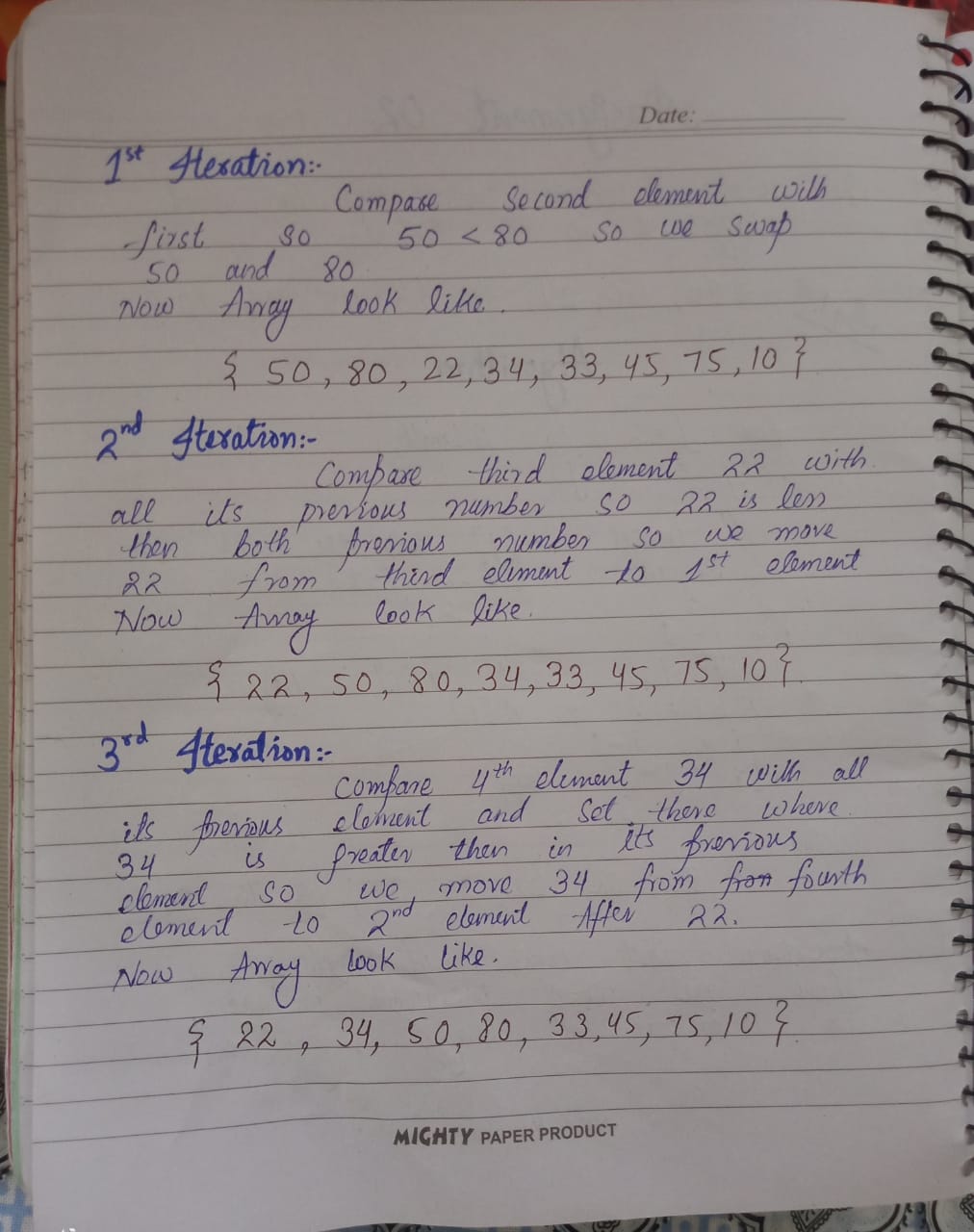
**ASSIGNMENT # 02 (CLO – 3)**

**Task # 01:** Write Insertion Sort Algorithm and apply it on the list {80, 50, 22, 34, 33, 45, 75, and 10}

**SOLUTION:** 



**Task # 02:** Identify the BIG O notation of Bubble sort and Insertion Sort Algorithm. Explain your answer.

**BIG O OF BUBBLE SORT:**

**Best Case:**

In the best case the array will be already sorted & if the numbers are already sorted

In ascending order the algorithm will determine in the first iteration that no number of

Pairs. Need to be swapped and will then terminate immediately.

The algorithm must perform (n-1) companions here.

The best-case Time complexity of Bubble Sort is 0 (n).

Example: {1, 2, 3, 4, 5, 63}

**Worst Case:**

In the worst case, the array will be in form of descending order. If the algorithm is in worst case form, so there will be (n-1) comparison in 1st phase, (n-2) in 2nd phase and so on. The no. of swap will be same as number of comparisons.

The total number of comparisons will be given given as:-

(N-1)+ (n-2) + (n-3) +……+3+2+1

Sum = n (n-1) = n2-n

The time complexity of worst - case time of Bubble Sort is O (n2)

**BIG O OF INSERTION SORT:**

**Best Case:**

The best-case time complexity of insertion sort algorithm is O (n)

Time complexity Meaning that the time taken to sort a list is proportional to the

Number of elements in the list. This is the case when the list is already in the correct

Order.

The total number of comparison will be (n-1).

Example: {6, 9, 10, 11, 15, 16}

**Worst Case:**

The worst case for insertion sort will occur when the inputs list is in descending order.

To insert the last element we need at most (n-1) comparison, and at most (n-1) swaps. To

Insert the second to the last element, we need at most (n-2) comparison and at most

(n-2) swaps, and so on.

The number of operation needed to perform insertion sort is

Therefore

2 x (1+2+3+.... (N-2)+ (n-1)).

The sum can be calculated as

(2(n-1) (n-1+1))/ 2 = n (n-1) => n2-n

The time complexity of worst Case time is O (n2).

**Task # 03:** Write program in java to generate pseudorandom number using linear congruential generator. Xn+1 = (3xn + 2) mod 13 with seed x0 = 1

**SOLUTION:**

Public class DS\_ASSI\_Q3 {

Static void generate Random (in seed Value, in mod, in multiplier, in increment, in []

PseudoNumbers, in noOfRandomTerms)

{

Pseudo Numbers [0] = seed Value;

for (in I = 1; I < noOfRandomTerms; i++) {

pseudoNumbers[i]= ((pseudoNumbers[i - 1] \* multiplier) + increment)% mod;

}

}

public static void main(String[] args)

{

in seedValue = 1;

in mod = 13;

in multiplier = 3;

in increment = 2;

in noOfRandomTerms = 15;

in[] pseudoNumbers = new in[noOfRandomTerms];

generateRandom(seedValue, mod, multiplier, increment,

pseudoNumbers,noOfRandomTerms);

System.out.println("The PseudoRandom Numbers are: ");

for (in i = 0; i < noOfRandomTerms; i++) {

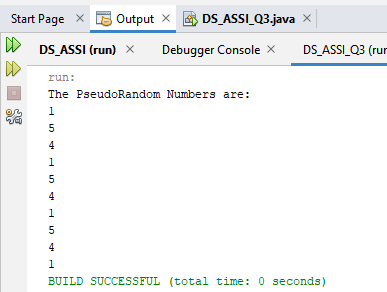
System.out.println(pseudoNumbers[i] + " ");

}

}

}

**OUTPUT:**



**Task # 04:** Write program that will use hash function to provide the memory location (without collision) on giving Enrollment no and memory address as input.

**SOLUTION:**

public static void main(String[] args) {

Scanner input = new Scanner (System.in);

Hashtable<Integer, Integer> h = new Hashtable<Integer, Integer>();

System.out.println("Enter the Number of total address: ");

in ad = input.nextInt();

for(in i=0; i < ad; i++){

System.out.println("Enter the Enrollment Number to Insert: ");

in en = input.nextInt();

in index = en%ad;

if(h.containsKey(index)){

System.out.println("COLLIOSION OCCURED !!!");

index = (index+1)%ad;

h.put(index, en);

}

else{

h.put(index, en);

}

System.out.println("Do you want to add again ? \nY OR N");

char ans = input.next().charAt(0);

if(ans=='y')

continue;

else

break;

}

//Formattting the HashTable

TreeMap<Integer, Integer> tm = new TreeMap<Integer, Integer>(h);

Set<Integer> keys = tm.keySet();

Iterator<Integer> itr = keys.iterator();

while (itr.hasNext()) {

Integer i = itr.next();

System.out.println("------------------");

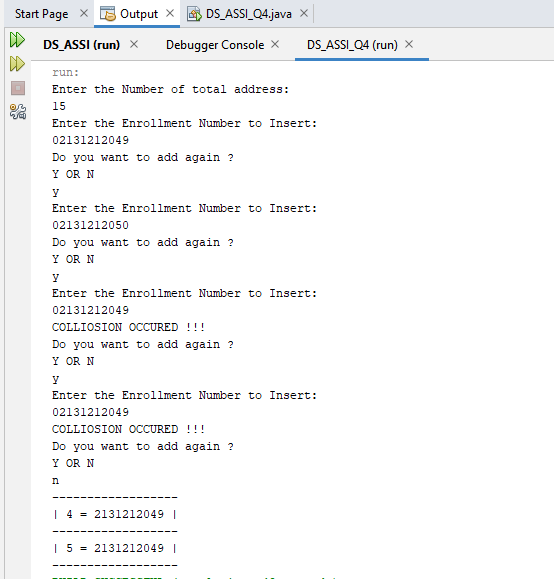
System.out.println("| "+ i + " = " + tm.get(i)+ " | ");

}

System.out.println("------------------");

}

**OUTPUT:**



**Task # 05:** Write program that takes a message and a positive integer k less than 26, encrypt this message using the shift cipher with key k; and given a message encrypted using a shift cipher with key k, decrypt this message.

**SOLUTION:**

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

char[] alphabet =

{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};

System.out.print("Enter what you want to choose ? \n 1 for Decryption \n 2 for Encryption\n ENTER : ");

in ans = input.nextInt();

switch(ans){

case 1:

System.out.println("Enter the Value of K: ");

in k = input.nextInt();

System.out.println("Enter what you want to Decrypt: ");

String s= input.next();

String str = s.toLowerCase();

System.out.println("\n");

char [] ch = str.toCharArray();

for (in i=0; i < ch.length; i++){

if( ch[i] == ' ' ) {

ch[i]=' ';

}

else{

for(in j=0; j<alphabet.length; j++){

if(ch[i]==alphabet[j]){

if(j<k){

ch[i]=alphabet[26-(k-j)];

}

else{

ch[i]=alphabet[(j-k)%26];

}}}} }

System.out.println("After Decrypting Answer is");

for( in i = 0; i < ch.length; i++ ){

if( ch[i] == ' ' ) {

System.out.println(' ');

}

else{

System.out.print(" "+ ch[i]);

}}

System.out.println("\n");

break;

case 2:

System.out.println("Enter the Value of 'K': ");

in k1 = input.nextInt();

System.out.println("Enter what you want to Encrypt: ");

String s2= input.next();

String str2 = s2.toLowerCase();

System.out.println("\n");

char [] ch2 = str2.toCharArray();

for (in i=0; i < ch2.length; i++){

if( ch2[i] == ' ' ) {

ch2[i]=' ';

}

else{

for(in j=0; j < alphabet.length; j++){

if(ch2[i]==alphabet[j]){

in jjj = j+k1;

ch2[i]=alphabet[jjj%26];

jjj=0;

break;

}}}}

System.out.println("After Encrypting Answer is: ");

for( in i = 0; i < ch2.length; i++ ){

if( ch2[i] == ' ' ) {

System.out.println(' ');

}

else{

System.out.print(" "+ ch2[i]);

}

}

System.out.println("\n");

break;

default:

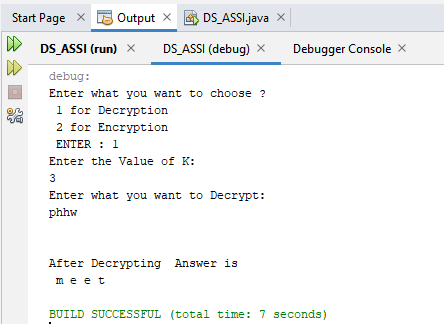
break;

}

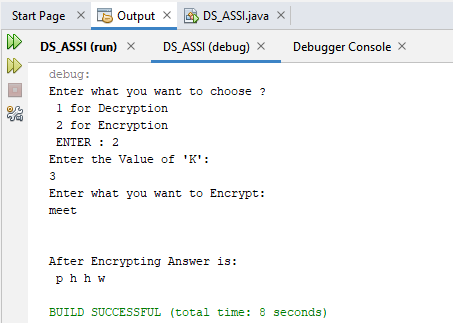
}

**OUTPUT:**

**DECRYPTION**

****

**ENCRYPTION**

****