

Laboratory Assignment File
for

Data Structure and Algorithms

Master of Technology
in
Computer Science & Engineering

Submitted by

Yugal
(Roll no. 25903053)



Department of Computer Science & Engineering
Dr. B R Ambedkar National Institute of Technology Jalandhar
Punjab, India-144008
May, 2026

Contents

Assignment 1: Basic Programs	28-01-2026	1
1.1 Hello World		1
1.2 Write a program to show Array		1
1.3 Write a program to show sum and average		1
Assignment 2: Sorting Algorithms	02-02-2026	3
2.1 Write a program to show Bubble Sort		3
2.2 Write a program to show Insertion Sort		3
2.3 Write a program to show Selection Sort		4
2.4 Write a program to show Merge Sort		5
2.5 Write a program to show Quick Sort		6
Assignment 3: String	05-02-2026	9
3.1 Write a program to Count Vowels in a String		9
3.2 Palindrome String		9
3.3 Remove Spaces from a String		10
3.4 Reverse a String		10
Assignment 4: Iteration	07-02-2026	12
4.1 Factorial using Iteration		12
Assignment 5: Searching	09-02-2026	13
5.1 Binary Search		13
5.2 Linear Search		16
Assignment 6: Recursion	11-02-2026	21
6.1 Factorial of a Number using Recursion		21
6.2 Fibonacci Series using Recursion		21
6.3 Palindrome String using Recursion		22
6.4 Power of a number using Recursion		23
6.5 Print numbers using Recursion		24
6.6 Recursion Factorial of a Number		24
6.7 Reverse Number using Recursion		25
6.8 String Recursion Menu		26
6.9 Sum of digits using Recursion		28
Assignment 7: Linked List	12-02-2026	30
7.1 Singly-Linked-List		30
7.2 Doubly-Linked-List		32

List of Figures

1	Program to show Array	1
2	Program to show sum and average	2
3	Program to show Bubble Sort	3
4	Program to show Insertion Sort	4
5	Program to show Selection Sort	5
6	Program to show Merge Sort	6
7	Program to show Quick Sort	8
8	Output of Count Vowels Program	9
9	Output of Palindrome Program	10
10	Output of Remove Spaces Program	10
11	Output of Reverse String Program	11
12	Output of Factorial using Iteration Program	12
13	Output of Binary Search Program	15
14	Output of Binary Search Program	16
15	Output of Linear Search Program	19
16	Output of Linear Search Program	20
17	Output of Factorial Program	21
18	Output of Fibonacci Program	22
19	Output of Palindrome Program	23
20	Output of Power of Number Program	24
21	Output of Print Numbers Program	24
22	Output of Recursion Factorial Program	25
23	Output of Reverse Number Program	26
24	Output of String Recursion Menu Program	27
25	Output of String Recursion Menu Program	28
26	Output of Sum of Digits Program	29
27	Singly Linked List Output	32
28	Output of Doubly Linked List Program	37

Assignment 1: Basic Programs

1.1 Hello World

```
1 public class Hello{  
2     public static void main(String arg[]){  
3         System.out.println("Hello");  
4     }  
5 }
```

1.2 Write a program to show Array

```
1 // Store and display the marks  
2 public class Array{  
3     public static void main(String... args){  
4         int[] marks = {70,75,80,90,95};  
5         for(int i=0;i<marks.length;i++){  
6             // System.out.println("Student Marks: "+marks[i]);  
7             System.out.println("Student " + (i+1) + ":" +marks[i]);  
8         }  
9     }  
10 }  
11 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/env /usr/lib/jvm/java-11-open  
jdk-amd64/bin/java -cp /home/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979c97b038c3  
955178/redhat.java/jdt_ws/DSA_1e4e73cd/bin Array  
Student 1:70  
Student 2:75  
Student 3:80  
Student 4:90  
Student 5:95  
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 1: Program to show Array

1.3 Write a program to show sum and average

```
1 // Sum and average of array elements  
2 class Sum_Average {  
3  
4     public static void main(String[] args){  
5         int[] a = {20,30,40,50,60};  
6         int sum = 0;
```

```
7
8     for(int x:a){
9         sum += x;
10    }
11    double average = (double) sum/a.length;
12
13    System.out.println("Sum: "+ sum);
14    System.out.println("Average: "+ average);
15}
16}
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/env /usr/lib/jvm/java-11-open
jdk-amd64/bin/java -cp /home/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979c97b038c3
955178/redhat.java/jdt_ws/DSA_1e4e73cd/bin Sum_Average
Sum: 200
Average: 40.0
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 2: Program to show sum and average

Assignment 2: Sorting Algorithms

2.1 Write a program to show Bubble Sort

```
1 // Sorting of array in Bubble Sort
2 public class BubbleSort{
3     public static void main(String[] args){
4         int arr[] = {5,1,4,2,8};
5         for(int i=0; i<arr.length-1; i++){
6             for(int j=0; j<arr.length-i-1; j++){
7                 if(arr[j]>arr[j+1]){
8                     int temp = arr[j];
9                     arr[j] = arr[j+1];
10                    arr[j+1] = temp;
11                }
12            }
13        }
14        for(int num : arr){
15            System.out.print(num + " ");
16        }
17    }
18 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979c97b038c3955178/redhat.java/jdt_ws/DSA_1e4e73cd/bin BubbleSort
◆ 1 2 4 5 8 (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 3: Program to show Bubble Sort

2.2 Write a program to show Insertion Sort

```
1 // Insertion Sort
2 public class InsertionSort {
3     public static void main(String[] args) {
4         int arr[] = {1,4,5,2,3,7,9,8,0,6};
5         for(int i = 1;i<arr.length; i++){
6             int key = arr[i];
7             int j = i-1;
8             while (j>=0 && arr[j]>key) {
9                 arr[j+1] = arr[j];
10                j--;
11            }
12        }
13    }
14 }
```

```

12         arr[j+1] = key;
13     }
14     for(int num : arr){
15         System.out.print(num + " ");
16     }
17 }
18 }
```

Output

```

● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/env /usr/lib/jvm/java-11-open
jdk-amd64/bin/java -cp /home/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979c97b038c3
955178/redhat.java/jdt_ws/DSA_1e4e73cd/bin InsertionSort
✉ 0 1 2 3 4 5 6 7 8 9 (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 4: Program to show Insertion Sort

2.3 Write a program to show Selection Sort

```

1 //Selection Sort
2 public class SelectionSort {
3     public static void main(String[] args) {
4         int arr[] = {3,5,2,6,-1,0,4,7,9,8,-2};
5
6         for(int i=0;i<arr.length-1;i++){
7             int minIndex = i;
8             for(int j = i+1; j<arr.length;j++){
9                 if(arr[j]<arr[minIndex]){
10                     minIndex = j;
11
12                 }
13             }
14             int temp = arr[minIndex];
15             arr[minIndex] = arr[i];
16             arr[i] = temp;
17         }
18         for(int num : arr){
19             System.out.print(num + " ");
20
21         }
22     }
23 }
24 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979c97b038c3955178/redhat.java/jdt_ws/DSA_1e4e73cd/bin SelectionSort
◆-2 -1 0 2 3 4 5 6 7 8 9 (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 5: Program to show Selection Sort

2.4 Write a program to show Merge Sort

```
1      //Merge Sort : Divide and Conquer
2  import java.util.Arrays;
3
4  public class MergeSort {
5
6      // Main merge sort function
7      public static void mergeSort(int[] arr, int left, int right){
8          if(left < right){
9              int mid = left + (right-left)/2;
10             // Sort Left Half
11             mergeSort(arr, left, mid);
12             // Sort Right Half
13             mergeSort(arr, mid+1, right);
14             // Merge Both Halves
15             merge(arr, left, mid, right);
16         }
17     }
18
19     // Merge two sorted subarrays
20     public static void merge(int[] arr, int left, int mid, int right){
21         int n1 = mid-left+1;
22         int n2 = right-mid;
23         int[] l = new int[n1];
24         int[] r = new int[n2];
25
26         // Copy data to temp arrays
27         for(int i=0; i<n1; i++)
28             l[i] = arr[left+i];
29         for(int j=0; j<n2; j++)
30             r[j] = arr[mid+1+j];
31
32         int i = 0, j=0, k=left;
33
34         // Merge temp arrays back into array/arr
35         while (i<n1 && j<n2) {
```

```

36         if  (l[i]<=r[j]) {
37             arr[k++] = l[i++];
38         }
39     else{
40         arr[k++] = r[j++];
41     }
42 }
43
44 // Copy remaining elements
45 while (i<n1) {
46     arr[k++] = l[i++];
47 }
48 while (j<n2) {
49     arr[k++] = r[j++];
50 }
51 }
52
53 // Driver Code
54 public static void main(String[] args) {
55
56     int[] arr = {38,27,43,3,9,82,10};
57     System.out.println("Before Sorting: ");
58     System.out.println(Arrays.toString(arr));
59
60     mergeSort(arr, 0, arr.length-1);
61
62     System.out.println("After Sorting: ");
63     System.out.println(Arrays.toString(arr));
64
65 }
66 }
```

Output

```

• (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ cd /home/yugal/Desktop/Sessions/DSA ;
/usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/yugal/.config/Code/User/works
paceStorage/13eba2f5617b89979c97b038c3955178/redhat.java/jdt_ws/DSA_1e4e73cd/bin MergeSort
Before Sorting:
[38, 27, 43, 3, 9, 82, 10]
After Sorting:
[3, 9, 10, 27, 38, 43, 82]
• (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 6: Program to show Merge Sort

2.5 Write a program to show Quick Sort

```

1   // Quick Sort : small -- pivot -- larger
2 import java.util.Arrays;
3
4 public class QuickSort {
5
6     // Main quick sort function
7     public static void quickSort(int[] arr, int low, int high){
8         if (low<high) {
9             int pivotIndex = partition(arr, low, high);
10
11             // Sort elements before and after partition
12             quickSort(arr, low, pivotIndex-1);
13             quickSort(arr, pivotIndex+1, high);
14         }
15     }
16
17     // Partition Function
18     public static int partition(int[] arr, int low, int high){
19         int pivot = arr[high]; // Choose last element as pivot
20         int i = low-1;
21         for(int j=low; j<high; j++){
22             if (arr[j]<pivot) {
23                 i++;
24
25                 // Swap arr[i] and arr[j]
26                 int temp = arr[i];
27                 arr[i] = arr[j];
28                 arr[j] = temp;
29             }
30         }
31
32         // Place pivot at correct position
33         int temp = arr[i+1];
34         arr[i+1] = arr[high];
35         arr[high] = temp;
36
37         return i+1;
38     }
39
40     // Driver code
41     public static void main(String[] args) {
42
43         int[] arr = {34,2,56,4,89,65,1,44};

```

```
44     System.out.println("Before Sorting: ");
45     System.out.println(Arrays.toString(arr));
46
47     quickSort(arr, 0, arr.length-1);
48     System.out.println("After Sorting: ");
49     System.out.println(Arrays.toString(arr));
50
51 }
52 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ cd /home/yugal/Desktop/Sessions/DSA ;
/usr/bin/env /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/yugal/.config/Code/User/works
paceStorage/13eba2f5617b89979c97b038c3955178/redhat.java/jdt_ws/DSA_1e4e73cd/bin QuickSort
Before Sorting:
[34, 2, 56, 4, 89, 65, 1, 44]
After Sorting:
[1, 2, 4, 34, 44, 56, 65, 89]
○ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 7: Program to show Quick Sort

Assignment 3: String

3.1 Write a program to Count Vowels in a String

```
1 package Strings;
2 // Count vowels name in String
3 public class CountVowels {
4     public static void main(String[] args) {
5         String str = "Eleven";
6         int count = 0;
7         for(int i=0; i<str.length();i++){
8             char ch = str.charAt(i);
9             if("AEIOUaeiou".indexOf(ch) != -1){
10                 count++;
11             }
12         }
13         System.out.println("Vowels: "+ count);
14     }
15 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ 
ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b8
.CountVowels
Vowels: 3
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 8: Output of Count Vowels Program

3.2 Palindrome String

```
1 package Strings;
2 // Word is spelled in same way forward or backward. Eg: NITIN, MADAM
3 //Palindrome
4 public class Palindrome {
5     public static void main(String[] args) {
6         String str = "nitin";
7         String rev = "";
8
9         for(int i=str.length()-1; i>=0;i--){
10             rev += str.charAt(i);
11         }
12     }
13 }
```

```

12
13     if (str.equals(rev)) {
14         System.out.println(str + " is a palindrome.");
15     } else {
16         System.out.println(str + " is not a palindrome.");
17     }
18 }
19 }
```

Output

- (base) **yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA\$**
`ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b
.Palindrome`
`nitin is a palindrome.`
- ◆ (base) **yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA\$**

Figure 9: Output of Palindrome Program

3.3 Remove Spaces from a String

```

1 package Strings;
2 // Remove spaces from string
3 public class RemoveSpaces {
4     public static void main(String[] args) {
5         String str = "Every Expert Was Once A Beginner.";
6         System.out.println(str.replace(" ", ""));
7     }
8 }
```

Output

- (base) **yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA\$** `c
/java-11-openjdk-amd64/bin/java -cp /home/yugal/.config/Cod
t.java/jdt_ws/DSA_1e4e73cd/bin Strings.RemoveSpaces`
`EveryExpertWasOnceABeginner.`
- (base) **yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA\$**

Figure 10: Output of Remove Spaces Program

3.4 Reverse a String

```
1 package Strings;
2
3 public class ReverseString {
4
5     public static void main(String[] args) {
6
7         String str = "Hello";
8         String rev = "";
9
10        for(int i = str.length()-1; i>=0; i--){
11            rev = rev + str.charAt(i);
12        }
13        System.out.println("Reversed String: "+ rev);
14    }
15 }
16 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$  
ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b8  
.ReverseString  
Reversed String: olleH  
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 11: Output of Reverse String Program

Assignment 4: Iteration

4.1 Factorial using Iteration

```
1 package Iteration;
2 // Factorial program using iteration
3 public class FactorialUsingIteration {
4     // Driver code
5     public static void main(String[] args) {
6         int n = 5;
7         int fact = 1;
8         for(int i=1; i<=n; i++){
9             fact *= i;
10        }
11        System.out.println("Factorial : "+fact);
12    }
13 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/en
amd64/bin/java -cp /home/yugal/.config/Code/User/workspaceStorage/13e
edhat.java/jdt_ws/DSA_1e4e73cd/bin Iteration.FactorialUsingIteration
Factorial : 120
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 12: Output of Factorial using Iteration Program

Assignment 5: Searching

5.1 Binary Search

```
1 // Binary Search
2 package Searching;
3 import java.util.Scanner;
4
5 public class BinarySearch {
6
7     //Driver Code
8     public static void main(String[] args) {
9         Scanner sc = new Scanner(System.in);
10        int[] arr = new int[20];// OK
11        int n = 0, choice, key;
12
13        do{
14            System.out.println("\n ---Binary Search---");
15            System.out.println("1. Insert Elements");
16            System.out.println("2. Display Elements");
17            System.out.println("3. Binary Search");
18            System.out.println("4. Exit");
19
20            System.out.print("Enter your choice: ");
21            choice = sc.nextInt();
22
23            switch (choice) {
24                case 1:
25                    System.out.print("Enter number of elements: ");
26                    n = sc.nextInt();
27                    System.out.println("Enter Elements: ");
28
29                    for(int i=0; i<n; i++){
30                        arr[i] = sc.nextInt();
31                    }
32                    break;
33
34                case 2:
35                    System.out.print("Array Elements: ");
36                    for(int i=0; i<n; i++){
37                        System.out.print(arr[i] + " ");
38                    }
39                    System.out.println();
40                    break;
41            }
42        } while(choice != 4);
43    }
44}
```

```

41
42     case 3:
43         System.out.print("Enter element to search: ");
44         key = sc.nextInt();
45         int low = 0, high = n-1;
46         boolean found = false;
47         while (low <= high) {
48             int mid = (low+high) / 2;
49             if (arr[mid] == key) {
50                 System.out.println("Element found at index:
51                             " + mid);
52                 found = true;
53                 break;
54             }
55             else if (arr[mid] < key) {
56                 low = mid+1;
57             }
58             else{
59                 high = mid-1;
60             }
61         }
62         if (!found)
63             System.out.println("Element not found");
64         break;
65
66     case 4:
67         System.out.println("Exiting Program");
68         break;
69
70     default:
71         System.out.println("Invalid Choice"); // break not
72         compulsory
73     }
74     while (choice != 4);
75     sc.close();
76 }
77 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/env /usr/lib/jvm/java-11-openJDK/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979c97b038c3955178/redhat.java/jdt_ng.BinarySearch

---Binary Search---
1. Insert Elements
2. Display Elements
3. Binary Search
4. Exit
Enter your choice: 1
Enter number of elements: 2
Enter Elements:
1
2

---Binary Search---
1. Insert Elements
2. Display Elements
3. Binary Search
4. Exit
Enter your choice: 2
Array Elements: 1 2

---Binary Search---
1. Insert Elements
2. Display Elements
3. Binary Search
4. Exit
Enter your choice: 3
```

Figure 13: Output of Binary Search Program

```
(base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/b
amd64/bin/java -cp /home/yugal/.config/Code/User/workspaceStorag
edhat.java/jdt_ws/DSA_1e4e73cd/bin Searching.BinarySearch

---Binary Search---
1. Insert Elements
2. Display Elements
3. Binary Search
4. Exit
Enter your choice: 2
Array Elements: 1 2

---Binary Search---
1. Insert Elements
2. Display Elements
3. Binary Search
4. Exit
Enter your choice: 3
Enter element to search: 2
Element found at index: 1

---Binary Search---
1. Insert Elements
2. Display Elements
3. Binary Search
4. Exit
Enter your choice: 4
Exiting Program
```

Figure 14: Output of Binary Search Program

5.2 Linear Search

```
1 package Searching;
2
3 import java.util.Scanner;
4
5 public class LinearSearch {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         int[] arr = new int[10];
9         int n = 0, choice, key;
10        boolean found;
11    }
```

```

12     do{
13         System.out.println("\n ---Linear Search Menu---");
14         System.out.println("1. Insert Elements");
15         System.out.println("2. Display Elements");
16         System.out.println("3. Linear Search");
17         System.out.println("4. Exit");
18
19         System.out.print("Enter your choice: ");
20         choice = sc.nextInt();
21
22         switch (choice) {
23             case 1:
24                 System.out.print("Enter number of elements: ");
25                 n = sc.nextInt();
26                 System.out.println("Enter Elements: ");
27
28                 for(int i=0; i<n; i++){
29                     arr[i] = sc.nextInt();
30                 }
31                 break;
32
33             case 2:
34                 System.out.print("Array Elements: ");
35                 for(int i=0; i<n; i++){
36                     System.out.print(arr[i] + " ");
37                 }
38                 System.out.println();
39                 break;
40
41             case 3:
42                 System.out.print("Enter element to search: ");
43                 key = sc.nextInt();
44                 found=false;
45
46                 for(int i=0; i<n; i++){
47                     if (arr[i] == key) {
48                         System.out.println("Element found at index:
49                             " + i);
50                         found = true;
51                         break;
52                     }
53                 }
54
55                 if (!found)

```

```
55     System.out.println("Element not found");
56     break;
57
58 case 4:
59     System.out.println("Exiting Program");
60     break;
61
62 default:
63     System.out.println("Invalid Choice");
64     break;
65 }
66 }
67
68 while (choice != 4);
69 sc.close();
70 }
71 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr  
ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979  
ng.LinearSearch  
  
---Linear Search Menu---  
1. Insert Elements  
2. Display Elements  
3. Linear Search  
4. Exit  
Enter your choice: 1  
Enter number of elements: 2  
Enter Elements:  
1  
2  
  
---Linear Search Menu---  
1. Insert Elements  
2. Display Elements  
3. Linear Search  
4. Exit  
Enter your choice: 2  
Array Elements: 1 2  
  
---Linear Search Menu---  
1. Insert Elements  
2. Display Elements  
3. Linear Search  
4. Exit  
Enter your choice: 3
```

Figure 15: Output of Linear Search Program

```
Enter element to search: 2
Element found at index: 1

---Linear Search Menu---
1. Insert Elements
2. Display Elements
3. Linear Search
4. Exit
Enter your choice: 4
Exiting Program
◆ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 16: Output of Linear Search Program

Assignment 6: Recursion

6.1 Factorial of a Number using Recursion

```
1 package Recursion;
2
3 import java.util.Scanner;
4
5 // Factorial Program --> eg: 4! = 4*3*2*1 = 24
6 public class Factorial {
7
8     static int factorial(int n){
9         if (n == 0)
10             return 1;
11         return n * factorial(n-1);
12     }
13
14     //Driver Code
15     public static void main(String[] args) {
16
17         System.out.println("Enter any number: ");
18         Scanner sc = new Scanner(System.in);
19         int num = sc.nextInt();
20         System.out.println("Factorial of " + num + " is " + factorial(num));
21         sc.close();
22     }
23 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /amd64/bin/java -cp /home/yugal/.config/Code/User/workspace$edhat.java/jdt_ws/DSA_1e4e73cd/bin Recursion.Factorial
Enter any number:
4
Factorial of 4 is 24
◆ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ █
```

Figure 17: Output of Factorial Program

6.2 Fibonacci Series using Recursion

```

1 package Recursion;
2 import java.util.Scanner;
3
4 //Fibonacci Series --> 0,1,0+1=1,1+1=2,2+1=3...
5 public class Fibonacci {
6     static int fibonacci(int n){
7         if (n == 0)
8             return 0;
9         if (n == 1)
10            return 1;
11         return fibonacci(n-1) + fibonacci(n-2);
12     }
13     //Driver Code
14     public static void main(String[] args) {
15
16         Scanner sc = new Scanner(System.in);
17         System.out.println("Enter Element: ");
18
19         int terms = sc.nextInt();
20         for(int i = 0; i<terms; i++){
21             System.out.print(fibonacci(i) + " ");
22             sc.close();
23         }
24     }
25 }
```

Output

```

● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ cd /home/yug
/java-11-openjdk-amd64/bin/java -cp /home/yugal/.config/Code/User/work
t.java/jdt_ws/DSA_1e4e73cd/bin Recursion.Fibonacci
Enter Element:
5
○ 0 1 1 2 3 (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ █
```

Figure 18: Output of Fibonacci Program

6.3 Palindrome String using Recursion

```

1 package Strings;
2 // Remove spaces from string
3 public class RemoveSpaces {
4     public static void main(String[] args) {
```

```

5     String str = "Every Expert Was Once A Beginner.";
6     System.out.println(str.replace(" ", ""));
7 }
8 }
```

Output

```

● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ 
ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b8
on.PalindromeRecursion
true
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 19: Output of Palindrome Program

6.4 Power of a number using Recursion

```

1 package Strings;
2
3 public class ReverseString {
4
5     public static void main(String[] args) {
6
7         String str = "Hello";
8         String rev = "";
9
10        for(int i = str.length()-1; i>=0; i--){
11            rev = rev + str.charAt(i);
12        }
13        System.out.println("Reversed String: "+ rev);
14    }
15 }
16 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/env /usr/home/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979c97b038c395511.on.PowerOfNumber
25
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ █
```

Figure 20: Output of Power of Number Program

6.5 Print numbers using Recursion

```
1 package Strings;
2
3 public class ReverseString {
4
5     public static void main(String[] args) {
6
7         String str = "Hello";
8         String rev = "";
9
10        for(int i = str.length()-1; i>=0; i--){
11            rev = rev + str.charAt(i);
12        }
13        System.out.println("Reversed String: "+ rev);
14    }
15}
16}
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ /usr/bin/env /usr/home/yugal/.config/Code/User/workspaceStorage/13eba2f5617b89979c97b038c395511.on.PrintNumbers
❖ 1 2 3 4 5 6 7 8 9 (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ █
```

Figure 21: Output of Print Numbers Program

6.6 Recursion Factorial of a Number

```
1 package Strings;
2
3 public class ReverseString {
```

```

4
5     public static void main(String[] args) {
6
7         String str = "Hello";
8         String rev = "";
9
10        for(int i = str.length()-1; i>=0; i--){
11            rev = rev + str.charAt(i);
12        }
13        System.out.println("Reversed String: "+ rev);
14    }
15}
16

```

Output

```

● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ 
ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b8
on.RecursiveFactorial
Factorial of 5 is : 120
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ 

```

Figure 22: Output of Recursion Factorial Program

6.7 Reverse Number using Recursion

```

1 package Strings;
2
3 public class ReverseString {
4
5     public static void main(String[] args) {
6
7         String str = "Hello";
8         String rev = "";
9
10        for(int i = str.length()-1; i>=0; i--){
11            rev = rev + str.charAt(i);
12        }
13        System.out.println("Reversed String: "+ rev);
14    }
15}
16

```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$  
ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b8  
on.ReversingNumber  
987654321  
◆ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$ █
```

Figure 23: Output of Reverse Number Program

6.8 String Recursion Menu

```
1 package Strings;  
2  
3 public class ReverseString {  
4  
5     public static void main(String[] args) {  
6  
7         String str = "Hello";  
8         String rev = "";  
9  
10        for(int i = str.length() -1; i >= 0; i--){  
11            rev = rev + str.charAt(i);  
12        }  
13        System.out.println("Reversed String: " + rev);  
14    }  
15}  
16}
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Ses  
ome/yugal/.config/Code/User/workspaceStorage/1  
on.StringRecursionMenu

    ---- STRING RECURSION MENU ----
1. Find Length
2. Reverse String
3. Check Palindrome
4. Count Vowels
5. Exit
Enter your choice: 1
Enter a string: yugal
5

    ---- STRING RECURSION MENU ----
1. Find Length
2. Reverse String
3. Check Palindrome
4. Count Vowels
5. Exit
Enter your choice: 2
Enter a string: yugal
laguy

    ---- STRING RECURSION MENU ----
1. Find Length
2. Reverse String
3. Check Palindrome
4. Count Vowels
```

Figure 24: Output of String Recursion Menu Program

```
5. Exit
Enter your choice: 3
Enter a string: nitin
true

-----STRING RECURSION MENU-----
1. Find Length
2. Reverse String
3. Check Palindrome
4. Count Vowels
5. Exit
Enter your choice: 4
Enter a string: hello
2

-----STRING RECURSION MENU-----
1. Find Length
2. Reverse String
3. Check Palindrome
4. Count Vowels
5. Exit
Enter your choice: 5
Program Closed
* (base) yugal@yugal-Inspiron-3583::
```

Figure 25: Output of String Recursion Menu Program

6.9 Sum of digits using Recursion

```
1 package Strings;
```

```
2
3 public class ReverseString {
4
5     public static void main(String[] args) {
6
7         String str = "Hello";
8         String rev = "";
9
10        for(int i = str.length() -1; i >= 0; i--){
11            rev = rev + str.charAt(i);
12        }
13        System.out.println("Reversed String: " + rev);
14
15    }
16}
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$  
amd64/bin/java -cp /home/yugal/.config/Code/User/workspace  
edhat.java/jdt_ws/DSA_le4e73cd/bin Recursion.SumOfDigits  
Sum = 10  
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 26: Output of Sum of Digits Program

Assignment 7: Linked List

7.1 Singly-Linked-List

```
1 public class SinglyLinkedList {
2     // Node class
3     class Node{
4         int data;
5         Node next;
6         Node(int data){
7             this.data = data;
8             this.next = null;
9         }
10    }
11
12    Node head = null;
13    // Insert at end
14    public void insertEnd(int data){
15        Node newNode = new Node(data);
16        if(head == null){
17            head = newNode;
18            return;
19        }
20
21        Node temp = head;
22        while(temp.next != null){
23            temp = temp.next;
24        }
25        temp.next = newNode;
26    }
27
28    // Insert at beginning
29    public void insertBeginning(int data){
30        Node newNode = new Node(data);
31        newNode.next = head;
32        head = newNode;
33    }
34
35    // Delete by value
36    public void delete(int key){
37        if (head == null)
38            return;
39        if (head.data == key) {
40            head = head.next;
```

```

41         return;
42     }
43
44     Node temp = head;
45     while (temp.next != null && temp.next.data != key) {
46         temp = temp.next;
47     }
48     if(temp.next != null){
49         temp.next = temp.next.next;
50     }
51 }
52
53 // Traverse
54 public void display(){
55     Node temp = head;
56     while(temp != null){
57         System.out.print(temp.data + " -> ");
58         temp = temp.next;
59     }
60
61     System.out.println("null");
62 }
63
64 // Driver code
65 public static void main(String[] args) {
66     SinglyLinkedList list = new SinglyLinkedList();
67     list.insertEnd(10);
68     list.insertEnd(20);
69     list.insertEnd(30);
70     System.out.println("---Insert End---");
71     list.display();
72     list.insertBeginning(1);
73     list.insertBeginning(2);
74     list.insertBeginning(3);
75     list.insertBeginning(4);
76     list.insertBeginning(5);
77     System.out.println("---Insert Beginning---");
78     list.display();
79
80     list.delete(30);
81     System.out.println("---Delete 30---");
82     list.display();
83 }
84

```

```
85 |
86 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$  
ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b:  
inkedList  
---Insert End---  
10->20->30->null  
---Insert Beginning---  
5->4->3->2->1->10->20->30->null  
---Delete 30---  
5->4->3->2->1->10->20->null  
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 27: Singly Linked List Output

7.2 Doubly-Linked-List

```
1 // Doubly Linked List Program
2 public class DoublyLinkedList {
3     // Node class
4     class Node{
5         int data;
6         Node prev; // Previous
7         Node next; //Next Pointer
8         Node(int data){
9             this.data = data;
10            this.prev = null;
11            this.next = null;
12        }
13    }
14    Node head = null;
15
16    // Insert at beginning
17    public void insertAtBeginning(int data){
18        Node newNode = new Node(data);
19        if(head == null){
20            head = newNode;
21            return;
22        }
```

```

23     newNode.next = head;
24     head.prev = newNode;
25     head = newNode;
26 }
27
28 // Insert at end
29 public void insertAtEnd(int data){
30     Node newNode = new Node(data);
31     if(head == null){
32         head = newNode;
33         return;
34     }
35     Node temp = head;
36     while (temp.next != null) {
37         temp = temp.next;
38     }
39     temp.next = newNode;
40     newNode.prev = temp;
41 }
42
43 // Insert at Position (1- Based Index)
44 public void insertAtPosition(int data, int position){
45     if (position <= 0) {
46         System.out.println("Invalid Position");
47         return;
48     }
49     if (position == 1) {
50         insertAtBeginning(data);
51         return;
52     }
53     Node newNode = new Node(data);
54     Node temp = head;
55
56     for(int i=1; temp != null && i<position-1; i++){
57         temp = temp.next;
58     }
59     if (temp == null) {
60         System.out.println("Position out of range");
61         return;
62     }
63     newNode.next = temp.next;
64     if (temp.next != null) {
65         temp.next.prev = newNode;
66     }

```

```

67     temp.next = newNode;
68     newNode.prev = temp;
69 }
70
71 // Delete from Beginning
72 public void deleteFromBeginning(){
73     if (head == null) {
74         System.out.println("List is Empty");
75         return;
76     }
77     head = head.next;
78     if (head != null) {
79         head.prev = null;
80     }
81 }
82
83 // Delete from End
84 public void deleteFromEnd(){
85     if (head == null) {
86         System.out.println("List is Empty");
87         return;
88     }
89     if (head.next == null) {
90         head = null;
91         return;
92     }
93     Node temp = head;
94     while (temp.next != null) {
95         temp = temp.next;
96     }
97     temp.prev.next = null;
98 }
99
100 // Delete from Position
101 public void deleteFromPosition(int position){
102     if (head == null || position <= 0) {
103         System.out.println("Invalid Operation");
104         return;
105     }
106     if (position == 1){
107         deleteFromBeginning();
108         return;
109     }
110     Node temp = head;

```

```

111     for(int i = 1; temp != null && i < position; i++){
112         temp = temp.next;
113     }
114     if (temp == null) {
115         System.out.println("Position out of Range.");
116         return;
117     }
118     if (temp.next != null) {
119         temp.next.prev = temp.prev;
120     }
121     if (temp.prev != null) {
122         temp.prev.next = temp.next;
123     }
124 }
125
126 // Search
127 public void search(int key){
128     Node temp = head;
129     int position = 1;
130     while (temp != null) {
131         if (temp.data == key) {
132             System.out.println("Element found at position: "+
133                             position);
134             return;
135         }
136         temp = temp.next;
137         position++;
138     }
139     System.out.println("Element Not Found!");
140 }
141
142 // Display forward
143 public void displayForward(){
144     Node temp = head;
145     System.out.print("Forward: ");
146     while (temp != null) {
147         System.out.print(temp.data + " ");
148         temp = temp.next;
149     }
150     System.out.println();
151 }
152 // Display Backward
153 public void displayBackward(){
154     if (head == null)

```

```

154         return;
155     Node temp = head;
156     while (temp.next != null) {
157         temp = temp.next;
158     }
159     System.out.print("Backward: ");
160     while (temp != null) {
161         System.out.print(temp.data + " ");
162         temp = temp.prev;
163     }
164     System.out.println();
165 }
166
167 // Driver code
168 public static void main(String[] args) {
169     DoublyLinkedList dll = new DoublyLinkedList();
170     dll.insertAtBeginning(10);
171     dll.insertAtBeginning(5);
172     dll.insertAtBeginning(50);
173     dll.insertAtBeginning(100);
174     dll.insertAtEnd(20);
175     dll.insertAtEnd(1000);
176     dll.insertAtEnd(2000);
177     dll.insertAtEnd(3000);
178     dll.insertAtEnd(25);
179     dll.insertAtPosition(15, 3);
180     dll.insertAtPosition(500, 1);
181     dll.displayForward();
182     dll.displayBackward();
183     dll.search(15);
184     dll.deleteFromBeginning();
185     dll.deleteFromEnd();
186     dll.deleteFromPosition(2);
187     dll.displayForward();
188     dll.displayBackward();
189 }
190 }
191 }
```

Output

```
● (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$  
ome/yugal/.config/Code/User/workspaceStorage/13eba2f5617b8  
inkedList  
Forward: 500 100 50 15 5 10 20 1000 2000 3000 25  
Backward: 25 3000 2000 1000 20 10 5 15 50 100 500  
Element found at position: 4  
Forward: 100 15 5 10 20 1000 2000 3000  
Backward: 3000 2000 1000 20 10 5 15 100  
❖ (base) yugal@yugal-Inspiron-3583:~/Desktop/Sessions/DSA$
```

Figure 28: Output of Doubly Linked List Program