Shivam Khaw
Div – B, Roll No.26

Sorting Techniques

Bubble sort

```
import java.util.Scanner;

public class bubbleSort {
	static int[] arr;
	int size;

	void insert() {

		System.out.println("Enter the size of the array : ");
		Scanner sc = new Scanner(System.in);
		size = sc.nextInt();
		arr = new int[size];
		for(int i = 0; i < size; i++) {
			System.out.println("Enter the element : ");
			int element = sc.nextInt();
			arr[i] = element;
		}
	}

	void bubblesort() {
		int i,j,temp;
		for(i = 0; i < size; i++) {
			for(j = 0; j < size - 1; j++) {
				if(arr[j] > arr[j+1]) {
					temp = arr[j];
					arr[j] = arr[j+1];
					arr[j+1] = temp;
				}
			}
		}

		System.out.println("Elements sorted");
	}

	void display() {
		for(int i = 0; i < arr.length; i++) {
			System.out.println(arr[i] + " ");
		}
	}
	public static void main(String[] args) {
		bubbleSort b1 = new bubbleSort();
		Scanner sc = new Scanner(System.in);
		int option;
		char ch = 'y';

		while(ch == 'y') {
```

```
                System.out.println("1. Insert Element");
                System.out.println("2. Sort array");
                System.out.println("3. Display array");

                System.out.println("Enter option : ");
                option = sc.nextInt();

                switch(option) {
                case 1 : {
                        b1.insert();
                        System.out.println("Do you want to continue ? (y / n)");
                        break;
                }
                case 2 :{
                        b1.bubblesort();
                        System.out.println("Do you want to continue ? (y / n)");
                        break;

                }
                case 3 : {
                        b1.display();
                        System.out.println("Do you want to continue ? (y / n)");
                        break;
                }
                }
                ch = sc.next().charAt(0);
            }
        }

}
```

Shivam Khaw
Div – B, Roll No.26

Shell sort

```
import java.util.Scanner;
public class shellSort {
	static int[] arr;
	int size;

	void insert() {

		System.out.println("Enter the size of the array : ");
		Scanner sc = new Scanner(System.in);
		size = sc.nextInt();
		arr = new int[size];
		for(int i = 0; i < size; i++) {
			System.out.println("Enter the element : ");
			int element = sc.nextInt();
			arr[i] = element;
		}
	}

	void shellsort() {
		int temp;
	   for(int gap = size / 2; gap > 0; gap /=2) {
		for(int i = gap; i< size;i++) {
			temp = arr[i];
			int j;
			for(j = i; j >= gap && arr[j-gap] > temp; j-= gap) {
				arr[j] = arr[j- gap];
			}

			arr[j] = temp;

		}
	   }

	   System.out.println("Elements sorted");
	}

	void display() {
		for(int i = 0; i < arr.length; i++) {
			System.out.println(arr[i] + " ");
		}
	}

	public static void main(String[] args) {
		shellSort b1 = new shellSort();
		Scanner sc = new Scanner(System.in);
```

```
		int option;
		char ch = 'y';

		while(ch == 'y') {
			System.out.println("1. Insert Element");
			System.out.println("2. Sort array");
			System.out.println("3. Display array");

			System.out.println("Enter option : ");
			option = sc.nextInt();

			switch(option) {
			case 1 : {
				b1.insert();
				System.out.println("Do you want to continue ? (y / n)");
				break;
			}
			case 2 :{
				b1.shellsort();
				System.out.println("Do you want to continue ? (y / n)");
				break;

			}
			case 3 : {
				b1.display();
				System.out.println("Do you want to continue ? (y / n)");
				break;
			}
			}
			ch = sc.next().charAt(0);
		}
	}

}
```

Shivam Khaw
Div – B, Roll No.26

Selection Sort

```
import java.util.Scanner;
public class selectionSort {
   static int[] arr;
   int size;
   void insert() {
      System.out.println("Enter the size of the array: ");
      Scanner sc = new Scanner(System.in);
      size = sc.nextInt();
      arr = new int[size];
      for(int i = 0; i < size; i++) {
         System.out.println("Enter the element at index " + i + ": ");
         int element = sc.nextInt();
         arr[i] = element;
      }
   }
   void display() {
      System.out.println("Array elements are: ");
      for(int i = 0; i < arr.length; i++) {
         System.out.print(arr[i] + " ");
      }
      System.out.println();
   }
  void selectionsort() {
      int i, j, temp;
      for(i = 0; i < size - 1; i++) { // Run till size-1 for optimization
         for(j = i + 1; j < size; j++) {
            if(arr[j] < arr[i]) {
               temp = arr[i];
               arr[i] = arr[j];
               arr[j] = temp;
            }
         }
      }
      System.out.println("Elements sorted successfully.");
   }
   public static void main(String[] args) {
      selectionSort b1 = new selectionSort();
      Scanner sc = new Scanner(System.in);
      int option;
      char ch = 'y';

      while(ch == 'y' || ch == 'Y') {  // Added support for both lowercase and uppercase 'Y'
         System.out.println("\nMenu:");
```

```
            System.out.println("1. Insert Elements");
            System.out.println("2. Sort Array");
            System.out.println("3. Display Array");
            System.out.println("Enter option: ");

            option = sc.nextInt();

            switch(option) {
               case 1:
                  b1.insert();
                  break;
               case 2:
                  if(arr != null) {
                     b1.selectionsort();
                  } else {
                     System.out.println("Array is empty. Please insert elements first.");
                  }
                  break;
               case 3:
                  if(arr != null) {
                     b1.display();
                  } else {
                     System.out.println("Array is empty. Please insert elements first.");
                  }
                  break;
               default:
                  System.out.println("Invalid option. Please select a valid option.");
                  break;
            }

            System.out.println("Do you want to continue? (y / n)");
            ch = sc.next().charAt(0);
         }

         sc.close(); // Close the scanner to prevent resource leaks
      }
   }
```

Shivam Khaw
Div – B, Roll No.26

Insertion Sort

```
package searching_technic;
import java.util.Scanner;
public class InsertionSort {
	static int[] arr;
	int size;
	void insert() {
		System.out.println("Enter the size of the array : ");
		Scanner sc = new Scanner(System.in);
		size = sc.nextInt();
		arr = new int[size];
		for(int i = 0; i < size; i++) {
			System.out.println("Enter the element : ");
			int element = sc.nextInt();
			arr[i] = element;
		}
	}
	void isort() {
		for(int i = 1; i < size; i++) {
			int key = arr[i];
			int j;
			for(j = i - 1; j >= 0 && arr[j] > key; j--) {
				arr[j + 1] = arr[j];
			}
			arr[j + 1] = key;
		}
		System.out.println("Elements sorted");
	}
	void display() {
		for(int i = 0; i < arr.length; i++) {
			System.out.print(arr[i] + " ");
		}
		System.out.println();
	}
	public static void main(String[] args) {
		InsertionSort b1 = new InsertionSort();
		Scanner sc = new Scanner(System.in);
		int option;
		char ch = 'y';

		while(ch == 'y' || ch == 'Y') {
			System.out.println("1. Insert Element");
			System.out.println("2. Sort array");
			System.out.println("3. Display array");
```

```
                System.out.println("Enter option : ");
                option = sc.nextInt();

                switch(option) {
                        case 1 : {
                                b1.insert();
                                System.out.println("Do you want to continue ? (y / n)");
                                break;
                        }
                        case 2 :{
                                b1.isort();
                                System.out.println("Do you want to continue ? (y / n)");
                                break;
                        }
                        case 3 : {
                                b1.display();
                                System.out.println("Do you want to continue ? (y / n)");
                                break;
                        }
                        default: {
                                System.out.println("Invalid option. Please try again.");
                        }
                }
                ch = sc.next().charAt(0);
        }
        sc.close();
    }
}
```

Shivam Khaw
Div – B, Roll No.26

Searching Technique

Linear search

```
package searching_technic;
import java.util.Scanner;
public class linearSearch {
	static int[] arr;
	int size;
	void insert() {
		System.out.println("Enter the size of the array : ");
		Scanner sc = new Scanner(System.in);
		size = sc.nextInt();
		arr = new int[size];
		for(int i = 0; i < size; i++) {
			System.out.println("Enter the element : ");
			int element = sc.nextInt();
			arr[i] = element;
		}
	}
	void lsearch() {
		Scanner sc = new Scanner(System.in);
		int num;
		System.out.println("Enter the number you want to search");
		num = sc.nextInt();
		int flag = 0;

		for(int i = 0; i < arr.length ; i++) {
			if(num == arr[i]) {
				flag = 1;
				break; // Break once the element is found
			}
		}

		if(flag == 1) {
			System.out.println(num+" is present in the array");
		}else {
			System.out.println("Element not found");
		}
	}

	public static void main(String[] args) {
		linearSearch b1 = new linearSearch();
		Scanner sc = new Scanner(System.in);
		int option;
		char ch = 'y';
```

```
		while(ch == 'y' || ch == 'Y') {
			System.out.println("1. Insert Element");
			System.out.println("2. Search Element");
			System.out.println("Enter option : ");
			option = sc.nextInt();

			switch(option) {
				case 1 : {
					b1.insert();
					System.out.println("Do you want to continue ? (y / n)");
					break;
				}
				case 2 :{
					b1.lsearch();
					System.out.println("Do you want to continue ? (y / n)");
					break;
				}
				default: {
					System.out.println("Invalid option. Please try again.");
				}
			}
			ch = sc.next().charAt(0);
		}
		sc.close();
	}
}
```

Shivam Khaw
Div – B, Roll No.26

Binary Search

```
package searching_technic;
import java.util.Scanner;
public class binarySearch {
	static int[] arr;
	int size;
	void insert() {
		System.out.println("Enter the size of the array : ");
		Scanner sc = new Scanner(System.in);
		size = sc.nextInt();
		arr = new int[size];
		for(int i = 0; i < size; i++) {
			System.out.println("Enter the element : ");
			int element = sc.nextInt();
			arr[i] = element;
		}
	}
	void selectionsort() {
		int i, j, temp;
		for(i = 0; i < size - 1; i++) { // Optimized outer loop
			for(j = i + 1; j < size; j++) {
				if(arr[j] < arr[i]) {
					temp = arr[i];
					arr[i] = arr[j];
					arr[j] = temp;
				}
			}
		}
	}
	void bsearch() {
		this.selectionsort();
		int l = 0, r = arr.length - 1, mid, target;
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the number you want to search");
		target = sc.nextInt();
		int flag = 0;
		while(l <= r) {
			mid = l + (r - l) / 2; // Prevents integer overflow
			if(arr[mid] == target) {
				flag = 1;
				break; // Break once the element is found
			}else if(arr[mid] > target) {
				r = mid - 1;
			}else {
```

```
                    l = mid + 1;
                }
            }
            if(flag == 1) {
                System.out.println("Element is there in the array");
            }else {
                System.out.println("Element not found");
            }
        }
        public static void main(String[] args) {
            binarySearch b1 = new binarySearch();
            Scanner sc = new Scanner(System.in);
            int option;
            char ch = 'y';

            while(ch == 'y' || ch == 'Y') {
                System.out.println("1. Insert Element");
                System.out.println("2. Search Element");
                System.out.println("Enter option : ");
                option = sc.nextInt();

                switch(option) {
                    case 1 : {
                        b1.insert();
                        System.out.println("Do you want to continue ? (y / n)");
                        break;
                    }
                    case 2 :{
                        b1.bsearch();
                        System.out.println("Do you want to continue ? (y / n)");
                        break;
                    }
                    default: {
                        System.out.println("Invalid option. Please try again.");
                    }
                }
                ch = sc.next().charAt(0);
            }
            sc.close();
        }
}
```

Shivam Khaw
Div – B, Roll No.26

LinkedList

Single linked list

```
package LinkedList;
import java.util.Scanner;
class snode {
  int  data;
  snode next;
  static snode p;
  void add(int num) {
     snode q = p;
     if (p == null) {
        p = new snode();
        p.data = num;
        p.next = null;
     } else {
        while (q.next != null) {
           q = q.next;
        }
        q.next = new snode();
        q.next.data = num;
        q.next.next = null;
     }
  }
  void display() {
     snode q = p;
     if (p == null) {
        System.out.println("No linked list");
     } else {
        while (q != null) {
           System.out.println(q.data);
           q = q.next;
        }
     }
  }
  int count() {
     int count = 0;
     snode q = p;
     while (q != null) {
        count++;
        q = q.next;
     }
     return count;
  }
```

Shivam Khaw
Div – B, Roll No.26

```
void sort() {
    snode i, j;
    int temp;
    for (i = p; i != null; i = i.next) {
        for (j = i.next; j != null; j = j.next) {
            if (i.data > j.data) {
                temp = i.data;
                i.data = j.data;
                j.data = temp;
            }
        }
    }
}
void insert(int pos, int num) {
    snode q = p;
    snode newNode = new snode();
    newNode.data = num;
    if (pos == 1) {
        newNode.next = p;
        p = newNode;
    } else if (pos == count() + 1) {
        add(num);
    } else {
        for (int i = 1; i < pos - 1; i++) {
            q = q.next;
        }
        newNode.next = q.next;
        q.next = newNode;
    }
}
void seqarch(int num) {
    int pos = 1;
    snode q = p;
    boolean found = false;
    while (q != null) {
        if (q.data == num) {
            System.out.println(num + " present at position = " + pos);
            found = true;
            break;
        }
        q = q.next;
        pos++;
    }
    if (!found) {
```

```
            System.out.println("'" + num + "' Not present");
        }
    }
    public static void main(String[] args) {
        int num;
        int pos;
        snode n = new snode();
        char ch = 'y';
        Scanner sc = new Scanner(System.in);
        while (ch == 'y') {
            System.out.println("1-Add");
            System.out.println("2-Display");
            System.out.println("3-Sort");
            System.out.println("4-Count");
            System.out.println("5-Insert");
            System.out.println("6-Search");
            System.out.print("Enter your option: ");
            int option = sc.nextInt();
            switch (option) {
                case 1:
                    System.out.print("Enter number: ");
                    num = sc.nextInt();
                    n.add(num);
                    break;
                case 2:
                    System.out.println("Displaying numbers:");
                    n.display();
                    break;
                case 3:
                    System.out.println("After sorting:");
                    n.sort();
                    n.display();
                    break;
                case 4:
                    System.out.println("Count of numbers: " + n.count());
                    break;
                case 5:
                    System.out.print("Enter the number you want to insert: ");
                    num = sc.nextInt();
                    System.out.print("Enter the position where you want to insert: ");
                    pos = sc.nextInt();
                    n.insert(pos, num);
                    break;
                case 6:
```

```
                System.out.print("Enter the number you want to search: ");
                num = sc.nextInt();
                n.seqarch(num);
                break;
            default:
                System.out.println("Invalid option");
        }
        System.out.print("Enter 'y' to continue: ");
        ch = sc.next().charAt(0);
    }
    sc.close();
  }
}
```

Shivam Khaw
Div – B, Roll No.26

Doubly linked list

```
package LinkedList;
import java.util.Scanner;
class dnode {
  int  data;
  dnode next;
  dnode prev;
  static dnode p;
  void add(int num) {
     dnode q = p;
     if (p == null) {
        p = new dnode();
        p.data = num;
        p.next = null;
        p.prev = null;
     } else {
        while (q.next != null) {
           q = q.next;
        }
        q.next = new dnode();
        q.next.data = num;
        q.next.next = null;
        q.next.prev = q;
     }
  }
  void display() {
     dnode q = p;
     if (p == null) {
        System.out.println("No linked list");
     } else {
        while (q != null) {
           System.out.println(q.data);
           q = q.next;
        }
     }
  }
  int count() {
     int count = 0;
     dnode q = p;
     while (q != null) {
        count++;
        q = q.next;
     }
     return count;
```

```
}
void sort() {
    dnode i, j;
    int temp;
    for (i = p; i != null; i = i.next) {
        for (j = i.next; j != null; j = j.next) {
            if (i.data > j.data) {
                temp = i.data;
                i.data = j.data;
                j.data = temp;
            }
        }
    }
}
void insert(int pos, int num) {
    dnode q = p;
    dnode newNode = new dnode();
    newNode.data = num;
    if (pos == 1) {
        newNode.next = p;
        if (p != null) {
            p.prev = newNode;
        }
        p = newNode;
        p.prev = null;
    } else if (pos == count() + 1) {
        add(num);
    } else {
        for (int i = 1; i < pos - 1; i++) {
            q = q.next;
        }
        newNode.next = q.next;
        newNode.prev = q;
        if (q.next != null) {
            q.next.prev = newNode;
        }
        q.next = newNode;
    }
}
void seqarch(int num) {
    int pos = 1;
    dnode q = p;
    boolean found = false;
    while (q != null) {
```

```
        if (q.data == num) {
            System.out.println(num + " present at position = " + pos);
            found = true;
            break;
        }
        q = q.next;
        pos++;
    }
    if (!found) {
        System.out.println("'" + num + "' Not present");
    }
}

void remove(int pos) {
    dnode temp;
    dnode q = p;
    if (p == null) {
        System.out.println("No Linked List to remove from.");
        return;
    }
    if (pos == 1) {
        p = p.next;
        if (p != null) {
            p.prev = null;
        }
    } else {
        for (int i = 1; i < pos - 1 && q != null; i++) {
            q = q.next;
        }
        temp = q.next;
        q.next = temp.next;
        if (temp.next != null) {
            temp.next.prev = q;
        }
    }
}
public static void main(String[] args) {
    int num;
    int pos;
    dnode n = new dnode();
    char ch = 'y';
    Scanner sc = new Scanner(System.in);
    while (ch == 'y') {
        System.out.println("1-Add");
```

```
System.out.println("2-Display");
System.out.println("3-Sort");
System.out.println("4-Count");
System.out.println("5-Insert");
System.out.println("6-Search");
System.out.println("7-remove");
System.out.print("Enter your option: ");
int option = sc.nextInt();
switch (option) {
   case 1:
      System.out.print("Enter number: ");
      num = sc.nextInt();
      n.add(num);
      break;
   case 2:
      System.out.println("Displaying numbers:");
      n.display();
      break;
   case 3:
      System.out.println("After sorting:");
      n.sort();
      n.display();
      break;
   case 4:
      System.out.println("Count of numbers: " + n.count());
      break;
   case 5:
      System.out.print("Enter the number you want to insert: ");
      num = sc.nextInt();
      System.out.print("Enter the position where you want to insert: ");
      pos = sc.nextInt();
      n.insert(pos, num);
      break;
   case 6:
      System.out.print("Enter the number you want to search: ");
      num = sc.nextInt();
      n.seqarch(num);
      break;

   case 7:
      System.out.print("Enter the possition number do you want to remove: ");
      pos = sc.nextInt();
      n.remove(pos);
      break;
```

```
            default:
                System.out.println("Invalid option");
        }
        System.out.print("Enter 'y' to continue: ");
        ch = sc.next().charAt(0);
    }
    sc.close();
  }
}
```

Shivam Khaw
Div – B, Roll No.26

Circular linked list

```
package LinkedList;
import java.util.Scanner;
class cinode {
  int  data;
  cinode next;
  cinode prev;
  static cinode p;
  void add(int num) {
     cinode newNode = new cinode();
     newNode.data = num;
     if (p == null) {
        p = newNode;
        p.next = p;
        p.prev = p;
     } else {
        cinode last = p.prev;
        last.next = newNode;
        newNode.prev = last;
        newNode.next = p;
        p.prev = newNode;
     }
  }
  void display() {
     if (p == null) {
        System.out.println("No linked list");
        return;
     }
     cinode q = p;
     do {
        System.out.println(q.data);
        q = q.next;
     } while (q != p);  // Loop until we're back at the head
  }
  int count() {
     if (p == null) return 0;
     int count = 0;
     cinode q = p;
     do {
        count++;
        q = q.next;
     } while (q != p);
     return count;
  }
```

Shivam Khaw
Div – B, Roll No.26

```
void sort() {
    if (p == null) return;
    cinode i = p;
    do {
        cinode j = i.next;
        while (j != p) {
            if (i.data > j.data) {
                int temp = i.data;
                i.data = j.data;
                j.data = temp;
            }
            j = j.next;
        }
        i = i.next;
    } while (i.next != p);
}
void insert(int pos, int num) {
    int size = count();
    if (pos < 1 || pos > size + 1) {
        System.out.println("Invalid position");
        return;
    }
    cinode newNode = new cinode();
    newNode.data = num;
    if (pos == 1) {
        if (p == null) {
            // Empty list case
            p = newNode;
            p.next = p;
            p.prev = p;
        } else {
            cinode last = p.prev;
            newNode.next = p;
            newNode.prev = last;
            last.next = newNode;
            p.prev = newNode;
            p = newNode; // Update head to new node
        }
    } else {
        cinode q = p;
        for (int i = 1; i < pos - 1; i++) {
            q = q.next;
        }
        newNode.next = q.next;
```

```
        newNode.prev = q;
        q.next.prev = newNode;
        q.next = newNode;
    }
}
void seqarch(int num) {
    if (p == null) {
        System.out.println("Linked list is empty");
        return;
    }
    int pos = 1;
    boolean found = false;
    cinode q = p;
    do {
        if (q.data == num) {
            System.out.println(num + " present at position = " + pos);
            found = true;
            break;
        }
        q = q.next;
        pos++;
    } while (q != p);
    if (!found) {
        System.out.println("'" + num + "' Not present");
    }
}
public static void main(String[] args) {
    int num;
    int pos;
    cinode n = new cinode();
    char ch = 'y';
    Scanner sc = new Scanner(System.in);
    while (ch == 'y') {
        System.out.println("1-Add");
        System.out.println("2-Display");
        System.out.println("3-Sort");
        System.out.println("4-Count");
        System.out.println("5-Insert");
        System.out.println("6-Search");
        System.out.print("Enter your option: ");
        int option = sc.nextInt();
        switch (option) {
            case 1:
                System.out.print("Enter number: ");
```

```
                num = sc.nextInt();
                n.add(num);
                break;
            case 2:
                System.out.println("Displaying numbers:");
                n.display();
                break;
            case 3:
                System.out.println("After sorting:");
                n.sort();
                n.display();
                break;
            case 4:
                System.out.println("Count of numbers: " + n.count());
                break;
            case 5:
                System.out.print("Enter the number you want to insert: ");
                num = sc.nextInt();
                System.out.print("Enter the position where you want to insert: ");
                pos = sc.nextInt();
                n.insert(pos, num);
                break;
            case 6:
                System.out.print("Enter the number you want to search: ");
                num = sc.nextInt();
                n.seqarch(num);
                break;
            default:
                System.out.println("Invalid option");
        }
        System.out.print("Enter 'y' to continue: ");
        ch = sc.next().charAt(0);
    }
    sc.close();
  }
}
```

Shivam Khaw
Div – B, Roll No.26

Stack and Queue

Stack using Doubly Linked List :

```
package Stack;
import java.util.Scanner;
class Stack {
  Stack prev;
  int data;
  Stack next;
  static Stack top;
  static Stack bottom;
  static void push(int num) {
     if (top == null && bottom == null) {
        Stack head = new Stack();
        head.data = num;
        bottom = top = head;
     } else {
        Stack temp = new Stack();
        temp.data = num;
        temp.prev = top;
        top.next = temp;
        top = temp;
     }
  }
  static void pop() {
     if (top != null) {
        if (top == bottom) { // Only one element in the stack
           top = bottom = null;
        } else {
           top = top.prev;
           top.next = null; // Nullify the next reference of the new top
        }
     } else {
        System.out.println("Stack is empty");
     }
  }
  static void popAll() {
     if (top != null) {
        while (top != bottom) {
           top = top.prev;
           top.next = null; // Nullify the next reference as we pop
        }
        top = bottom = null;
        System.out.println("Stack is now empty");
     } else {
        System.out.println("Stack is already empty");
     }
  }
```

Shivam Khaw
Div – B, Roll No.26

```
// Peek all elements in the stack
static void peekAll() {
    Stack temp = top;
    if (temp != null) {
        while (temp != null) {
            System.out.println(temp.data);
            temp = temp.prev;
        }
    } else {
        System.out.println("Stack is empty");
    }
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    char a = 'y';
    int z;
    while (a == 'y') {
        System.out.println("1. For push");
        System.out.println("2. For peekAll");
        System.out.println("3. For pop");
        System.out.println("4. For popAll");
        System.out.print("Enter Your Option: ");
        z = sc.nextInt();
        switch (z) {
            case 1:
                System.out.print("Enter the data: ");
                int p = sc.nextInt();
                push(p);
                break;
            case 2:
                peekAll();
                break;
            case 3:
                pop();
                peekAll();
                break;
            case 4:
                popAll();
                break;
            default:
                System.out.println("Invalid option. Try again.");
                break;
        }
        System.out.print("Do you want to continue? (y/n): ");
        a = sc.next().charAt(0);
    }
    sc.close();
}
```

Shivam Khaw
Div – B, Roll No.26

Stack Using Array

```
import java.util.*;
public class stackUsingArray{
        static int[] stack;
        int size;
        int top = -1;
        void push() {
                Scanner sc = new Scanner(System.in);
                if(top == -1) {
                        System.out.println("enter the size of stack");
                        size = sc.nextInt();
                        stack = new int[size];
                        top++;
                        System.out.println("Enter the number you want to push");
                        int num = sc.nextInt();
                        stack[top] = num;
                }else {
                        top++;
                        System.out.println("Enter the number you want to push");
                        int num = sc.nextInt();
                        stack[top] = num;
                }
        }

        public void pop() {
    if (top == -1) {
       System.out.println("Stack is empty. Cannot pop");
    } else {
       System.out.println("Popped " + stack[top] + " from stack");
       top--;
    }
  }
        public void display() {
    if (top == -1) {
       System.out.println("Stack is empty.");
    } else {
       System.out.print("Stack elements: ");
       for (int i = 0; i <= top; i++) {
          System.out.print(stack[i] + " ");
       }
       System.out.println();
    }
  }
```

Shivam Khaw
Div – B, Roll No.26

```
public static void main(String[] args) {
                    stackUsingArray n1 = new stackUsingArray();
            Scanner sc = new Scanner(System.in);

            int option;
            char ch='y';

            while(ch == 'y') {
                    System.out.println("1.Push");
                System.out.println("2.Pop");
                System.out.println("3.Display");

                System.out.println("Enter an Option");
                option = sc.nextInt();

                switch(option) {
                case 1:{
                    n1.push();
                    System.out.println("Do you want to continue? ");
          break;
                }
                case 2:{
                    n1.pop();
                    System.out.println("Do you want to continue? ");
          break;
                }
                case 3:{
                    n1.display();
                    System.out.println("Do you want to continue? ");
          break;

                }
                }

                ch = sc.next().charAt(0);

            }

        }
}
```

Shivam Khaw
Div – B, Roll No.26

Queue

Simple Queue

```
package queue;
import java.util.Scanner;
class queue {
  int data;
  queue next;
  static queue front;
  static queue rear;
  void insert(int num) {
    if ((front == null) && (rear == null)) {
      front = rear = new queue();
      front.data = num;
      front.next = null;
    } else {
      rear.next = new queue();
      rear.next.data = num;
      rear.next.next = null;
      rear = rear.next;
    }
  }
  void display() {
    queue q = front;
    if ((front == null) && (rear == null)) {
      System.out.println("No QUEUE");
    } else {
      while (q != null) {
        System.out.println(q.data);
        q = q.next;
      }
    }
  }
  void remove() {
    int num;
    if ((front == null) && (rear == null)) {
      System.out.println("Queue is empty");
    } else {
      num = front.data;
      if (front.next != null) {
        front = front.next;
      } else {
        front = null;
        rear = null;
      }
```

```
        System.out.println("Removed Element is: " + num);
    }
}
public static void main(String args[]) {
    queue n1 = new queue();
    Scanner sc = new Scanner(System.in);
    int num, option;
    char ch = 'y';
    while (ch == 'y') {
        System.out.println("1. Insert");
        System.out.println("2. Display");
        System.out.println("3. Remove");
        System.out.print("Enter an option: ");
        option = sc.nextInt();
        switch (option) {
            case 1:
                System.out.print("Enter Number: ");
                num = sc.nextInt();
                n1.insert(num);
                System.out.println("Number inserted successfully.");
                break;
            case 2:
                System.out.println("Displayed data is:");
                n1.display();
                break;
            case 3:
                n1.remove();
                break;
            default:
                System.out.println("Invalid option. Please try again.");
        }
        System.out.print("Do you want to continue? (y/n): ");
        ch = sc.next().charAt(0);
    }
    sc.close();
} }
```

Shivam Khaw
Div – B, Roll No.26

Doubly Ended Queue

Code:-

```
import java.util.Scanner;
class queue {
  int data;
  queue next;
  queue prev;
  static queue front;
  static queue rear;
  void insert(int num) {
     if ((front == null) && (rear == null)) {
        front = rear = new queue();
        front.prev = null;
        front.data = num;
        front.next = null;
     } else {
        rear.next = new queue();
        rear.next.prev = rear;
        rear.next.data = num;
        rear.next.next = null;
        rear = rear.next;
     }
  }
    void insert_f(int num) {
       queue temp = front;
     if ((front == null) && (rear == null)) {
        front = rear = new queue();
        front.prev = null;
        front.data = num;
        front.next = null;

     } else {

        temp =front;
        front = new queue();
        front.data = num;
        front.prev = null;
        front.next = temp;
     }
  }
```

Shivam Khaw
Div – B, Roll No.26

```
void display() {
    queue q = front;
    if ((front == null) && (rear == null)) {
        System.out.println("No QUEUE");
    } else {
        while (q != null) {
            System.out.println(q.data);
            q = q.next;
        }
    }
}
void remove() {
    int num;
    if ((front == null) && (rear == null)) {
        System.out.println("Queue is empty");
    } else {
        num = front.data;
      front=front.next;
      if(front==null)
      {
          rear=null;
      }
        System.out.println("Removed Element is: " + num);
    }
}
void remove_r() {
    int num;
    if ((front == null) && (rear == null)) {
        System.out.println("Queue is empty");
    } else {
        num = rear.data;
        rear = rear.next;
        if (rear== null) {
             front = null;
        }
        System.out.println("Removed Element is: " + num);
    }
}
public static void main(String args[]) {
    queue n1 = new queue();
    Scanner sc = new Scanner(System.in);
    int num, option;
    char ch = 'y';
    while (ch == 'y') {
```

```
        System.out.println("1. Insert");
        System.out.println("2. Display");
        System.out.println("3. Remove");
        System.out.println("4. Insert number from front");
        System.out.println("5. Remove_R");
        System.out.print("Enter an option: ");


        option = sc.nextInt();
        switch (option) {
            case 1:
                System.out.print("Enter Number: ");
                num = sc.nextInt();
                n1.insert(num);
                System.out.println("Number inserted successfully.");
                break;
            case 2:
                System.out.println("Displayed data is:");
                n1.display();
                break;
            case 3:
                n1.remove();
                break;

            case 4:
                System.out.print("Enter Number : ");
                num =  sc.nextInt();
                n1.insert_f(num);
                System.out.println("Number inserted successfully.");
                break;
            case 5:
                n1.remove_r();
                break;
            default:
                    System.out.println("Invalid option. Please try again.");

        }
        System.out.print("Do you want to continue? (y/n): ");
        ch = sc.next().charAt(0);
    }
    sc.close();
  }
}
```

Shivam Khaw
Div – B, Roll No.26

Priority queue (Data as a Priority) :

```
import java.util.*;
public class queue
{
   queue next;
   static queue front;
   static queue rear;
   int data;

   void insert(int num){
      if(front == null && rear == null){
         front = rear = new queue();
         front.data = num;
         front.next = null;
      }else{
         rear.next = new queue();
         rear.next.data = num;
         rear.next.next = null;
         rear = rear.next;

      }
   }
   void display(){
      queue q = front;
      if((front==null)&&(rear==null))
      {
        System.out.println("Queue is Empty");
      }
      else
      {
      while(q != null)
      {
         System.out.println(q.data);
         q = q.next;
      }
      }
   }
   void remove(){
      if(front == null && rear == null){
         System.out.println("Queue is empty");
      }else{
         int num = front.data;
         if(front.next != null)
         {
```

```
            front = front.next;
        }else{
            front = null;
            rear = null;
        }

        System.out.println("The removed number is " + num);
    }
}

void priority()
{
    Scanner sc = new Scanner(System.in);
    int num;
    if (front.data>=rear.data)
    {
       remove();
    }
    else
    {
        System.out.println("Enter number");
        num= sc.nextInt();
                insert(num);
    }
}

        public static void main(String[] args) {
                    queue n1 = new queue();

            Scanner sc = new Scanner(System.in);

            int num, option;
            char ch='y';

            while(ch =='y'){
                System.out.println("1.Insert");
                System.out.println("2.Display");
                System.out.println("3.remove");
                  System.out.println("4.Priority");

                System.out.println("Enter an Option");

                option= sc.nextInt();
```

```
            switch(option){
                case 1:
                {
                    System.out.println("Enter number");
                    num= sc.nextInt();
                    n1.insert(num);
                    System.out.println("Do you want to continue? ");
                    break;

                }
                case 2:
                {
                    n1.display();
                    System.out.println("Do you want to continue? ");
                    break;
                }
                case 3:
                {
                    n1.remove();
                    System.out.println("Do you want to continue? ");
                    break;
                }
                case 4:
                {
                    n1.priority();
                    System.out.println("Do you want to continue? ");
                    break;
                }
            }
            ch = sc.next().charAt(0);


        }
    }
}
```

Shivam Khaw
Div – B, Roll No.26

Priority queue with priority number

```
import java.util.*;
public class queue
{
   queue next;
   static queue front;
   static queue rear;
   int data;

   void insert(int num){
      if(front == null && rear == null){
         front = rear = new queue();
         front.data = num;
         front.next = null;
      }else{
         rear.next = new queue();
         rear.next.data = num;
         rear.next.next = null;
         rear = rear.next;

      }
   }
   void display(){
      queue q = front;
      if((front==null)&&(rear==null))
      {
        System.out.println("Queue is Empty");
      }
      else
      {
      while(q != null)
      {
         System.out.println(q.data);
         q = q.next;
      }
      }
   }
   void remove(){
      if(front == null && rear == null){
         System.out.println("Queue is empty");
      }else{
         int num = front.data;
         if(front.next != null)
         {
```

```
            front = front.next;
        }else{
            front = null;
            rear = null;
        }

        System.out.println("The removed number is " + num);
    }
}

void priority()
{
    Scanner sc = new Scanner(System.in);
    int num;
    if (front.data>=rear.data)
    {
       remove();
    }
    else
    {
        System.out.println("Enter number");
        num= sc.nextInt();
                insert(num);
    }
}
        public static void main(String[] args) {
                    queue n1 = new queue();
            Scanner sc = new Scanner(System.in);
            int num, option;
            char ch='y';
            while(ch =='y'){
                System.out.println("1.Insert");
                System.out.println("2.Display");
                System.out.println("3.remove");
                  System.out.println("4.Priority");
                System.out.println("Enter an Option");

                option= sc.nextInt();

                switch(option){
                    case 1:
                    {
                        System.out.println("Enter number");
                        num= sc.nextInt();
```

```
                    n1.insert(num);
                    System.out.println("Do you want to continue? ");
                    break;

                }
                case 2:
                {
                    n1.display();
                    System.out.println("Do you want to continue? ");
                    break;
                }
                case 3:
                {
                    n1.remove();
                    System.out.println("Do you want to continue? ");
                    break;
                }
                case 4:
                {
                    n1.priority();
                    System.out.println("Do you want to continue? ");
                    break;
                }
            }
            ch = sc.next().charAt(0);


        }
    }
}
```

Shivam Khaw
Div – B, Roll No.26

Circular Queue

```
package Cqueue;
import java.util.Scanner;
class CirQueue {
  int data;
  CirQueue next;
  static CirQueue front;
  static  CirQueue  rear;
  void insert(int num) {
     CirQueue newNode = new CirQueue();
     newNode.data = num;
     if (front == null) { // Queue is empty
        front = rear = newNode;
        rear.next = front; // Point rear to front to make it circular
     } else {
        rear.next = newNode;
        rear = newNode;
        rear.next = front; // Maintain the circular link
     }
  }
  void display() {
     if (front == null) {
        System.out.println("No QUEUE");
        return;
     }
     CirQueue q = front;
     do {
        System.out.println(q.data);
        q = q.next;
     } while (q != front);
  }
  void remove() {
     if (front == null) { // Queue is empty
        System.out.println("Queue is empty");
     } else if (front == rear) { // Only one element in the queue
        System.out.println("Removed Element is: " + front.data);
        front = rear = null;
     } else { // More than one element in the queue
        System.out.println("Removed Element is: " + front.data);
        front = front.next;
        rear.next = front; // Maintain the circular link
     }
  }
```

Shivam Khaw
Div – B, Roll No.26

```
  public static void main(String args[]) {
    CirQueue n1 = new CirQueue();
    Scanner sc = new Scanner(System.in);
    int num, option;
    char ch = 'y';
    while (ch == 'y' || ch == 'Y') {
      System.out.println("1. Insert");
      System.out.println("2. Display");
      System.out.println("3. Remove");
      System.out.print("Enter an option: ");
      option = sc.nextInt();
      switch (option) {
        case 1:
          System.out.print("Enter Number: ");
          num = sc.nextInt();
          n1.insert(num);
          System.out.println("Number inserted successfully.");
          break;
        case 2:
          System.out.println("Displayed data is:");
          n1.display();
          break;
        case 3:
          n1.remove();
          break;
        default:
          System.out.println("Invalid option. Please try again.");
      }
      System.out.print("Do you want to continue? (y/n): ");
      ch = sc.next().charAt(0);
    }
    sc.close();
  }
}
```

Shivam Khaw
Div – B, Roll No.26

Tree

Binary search tree

Code:-

```
package Tree;
import java.util.Scanner;
public class bst {
  int data;
  bst right;
  bst left;
  static bst root;

  void add(int num, bst q) {
    if (root == null) {
      root = new bst();
      root.data = num;
      root.left = null;
      root.right = null;
    } else {
      if (num < q.data) {
        if (q.left == null) {
          q.left = new bst();
          q.left.data = num;
          q.left.left = null;
          q.left.right = null;
        } else {
          add(num, q.left);
        }
      } else {
        if (q.right == null) {
          q.right = new bst();
          q.right.data = num;
          q.right.left = null;
          q.right.right = null;
        } else {
          add(num, q.right);
        }
      }
    }
  }
  void preorder(bst q) {
    if (q != null) {
      System.out.println(q.data);
      preorder(q.left);
      preorder(q.right);
```

```
    }
}
void inorder(bst q) {
    if (q != null) {
        inorder(q.left);
        System.out.println(q.data);
        inorder(q.right);
    }
}
void postorder(bst q) {
    if (q != null) {
        postorder(q.left);
        postorder(q.right);
        System.out.println(q.data);
    }
}
int count(bst q) {
    if (q == null) {
        return 0;
    }
    return 1 + count(q.left) + count(q.right);
}
boolean search(bst q, int num) {
    if (q == null) {
        return false;
    }
    if (q.data == num) {
        return true;
    } else if (num < q.data) {
        return search(q.left, num);
    } else {
          return search(q.right, num);
    }
}
public static void main(String[] args) {
    bst b = new bst();
    char ch = 'y';
    Scanner sc = new Scanner(System.in);
    while (ch == 'y') {
        System.out.println("1-Add");
        System.out.println("2-Preorder");
        System.out.println("3-Inorder");
        System.out.println("4-Postorder");
        System.out.println("5-Count");
```

```
        System.out.println("6-Search");
        System.out.print("Enter your option: ");
        int option = sc.nextInt();
        switch (option) {
            case 1:
                System.out.print("Enter number: ");
                int num = sc.nextInt();
                b.add(num, root);
                break;
            case 2:
                System.out.println("Preorder Traversal:");
                b.preorder(root);
                break;
            case 3:
                System.out.println("Inorder Traversal:");
                b.inorder(root);
                break;
            case 4:
                System.out.println("Postorder Traversal:");
                b.postorder(root);
                break;
            case 5:
                System.out.println("Count of nodes: " + b.count(root));
                break;

            case 6:
                System.out.print("Enter number to search: ");
                num = sc.nextInt();
                if (b.search(root, num)) {
                    System.out.println(num + " found in the tree.");
                } else {
                    System.out.println(num + " not found in the tree.");
                }
                break;

            default:
                System.out.println("Invalid option");
        }
        System.out.print("Enter 'y' to continue: ");
        ch = sc.next().charAt(0);
    }
    sc.close();
}
```

Shivam Khaw
Div – B, Roll No.26

Heap tree

```
public class heap {
        static class MaxHeap {
    private int[] Heap;
    private int size;
    private int maxsize;

    public MaxHeap(int size) {
       this.maxsize = size;
       this.size = 0;
       Heap = new int[this.maxsize ];
      Heap[0] = Integer.MAX_VALUE;
    }

    private int parent(int pos) {
       return pos / 2;
    }

    private int leftChild(int pos) {
       return (2 * pos)  ;
    }

    private int rightChild(int pos) {
       return (2 * pos) + 1;
    }


    private void swap(int fpos, int spos) {
       int tmp;
       tmp = Heap[fpos];
       Heap[fpos] = Heap[spos];
       Heap[spos] = tmp;
    }

    private void downHeapify(int pos) {
       if (pos >= (size / 2) && pos <= size)
          return;

       if (Heap[pos] < Heap[leftChild(pos)] ||
             Heap[pos] < Heap[rightChild(pos)]) {

          if (Heap[leftChild(pos)] > Heap[rightChild(pos)]) {
             swap(pos, leftChild(pos));
             downHeapify(leftChild(pos));
```

```
            } else {
                swap(pos, rightChild(pos));
                downHeapify(rightChild(pos));
            }
        }
    }
    private void heapifyUp(int pos) {
        int temp = Heap[pos];
        while(pos>0 && temp > Heap[parent(pos)]){
            Heap[pos] = Heap[parent(pos)];
            pos = parent(pos);
        }
        Heap[pos] = temp;
    }


    public void insert(int element) {
        Heap[++size] = element;


        int current = size;
        heapifyUp(current);

    }

    public void print() {
        for (int i = 1; i <= size / 2; i++) {
            int left = leftChild(i);
            int right = rightChild(i);
            String leftValue = (left <= size) ? String.valueOf(Heap[left]) : "null";
            String rightValue = (right <= size) ? String.valueOf(Heap[right]) : "null";

            System.out.println(Heap[i] + ": L- " + leftValue + " R- " + rightValue);
        }
    }

    public int extractMax() {
        int max = Heap[1];
        Heap[1] = Heap[size--];
        downHeapify(1);
        return max;
    }
}
```

```
        public static void main(String[] arg)
    {

        MaxHeap maxHeap = new MaxHeap(6);
        maxHeap.insert(12);
        maxHeap.insert(32);
        maxHeap.insert(1);
        maxHeap.insert(40);
        int s = maxHeap.size;

        maxHeap.print();
        System.out.println("The max is " + maxHeap.extractMax());
    }

}
```

Shivam Khaw
Div – B, Roll No.26

Hashing Technique

Hash

```
import java.util.Scanner;
public class hash {
	private long[] arr = new long[20];

	public hash() {
		for(int i = 0; i < 20; i++) {
			arr[i] = 0;
		}
	}
	void directMethod() {
		int address, key;
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter key");
		key = sc.nextInt();

		if(key >= 1 && key < 20) {
			address = key;
			arr[address] = key;
			System.out.println("Key entered");
		}else {
			System.out.println("Invalid Key");
		}
	}
	void subtractionMethod() {
		int address, key;
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter key");
		key = sc.nextInt();
		if(key >= 81 && key < 99) {
			address =100- key;
			arr[address] = key;
			System.out.println("Key entered");
		}else {
			System.out.println("Invalid Key");
		}
	}
	void ModuloDivisionMethod() {
		int address, key;
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter key");
		key =  sc.nextInt();
		address =( key % 20 ) + 1;
		arr[address] = key;
		System.out.println("Key entered");
	}
```

```
void ModuloDivisionWithCollision() {
        int address, key;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter key");
        key = sc.nextInt();

        address =( key % 20 ) + 1;
        if(arr[address] == 0) {
                arr[address] = key;
                System.out.println("Key entered");
        }else {
                arr[address + 1] = key;
                System.out.println("Key entered");
        }
}

void digitExtractionMethod() {
        int address, key, temp;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter four digit key");
        key = sc.nextInt();

        temp = (key / 10);
        address = ( temp % 20 ) + 1;
        arr[address] = key;
        System.out.println("Key entered");

}

void digitExtractionMethodwithCollision() {
        int address, key, temp;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter four digit key");
        key = sc.nextInt();

        temp = (key / 10) + 1;
        address = ( temp % 20 ) + 1;

        if(arr[address] == 0) {
                arr[address] = key;
                System.out.println("Key entered");
        }else {
                arr[address + 1] = key;
                System.out.println("Key entered");
        }
}

void display() {
```

```
            for(int i = 0; i< 20; i++) {
                        System.out.println("Arr[" + i + "]" + arr[i]);
            }
}



public static void main(String[] args) {
            hash h = new hash();
    Scanner sc = new Scanner(System.in);
    int key;
    int option;
    char ch='y';

    while(ch =='y'){
        System.out.println("1.direct Method");
        System.out.println("2.Subtraction Method");
        System.out.println("3.Modulo Division Method");
        System.out.println("4.Modulo Division With Collision");
        System.out.println("5.Digit Extraction Method");
        System.out.println("6.Digit Extraction Method with Collision");
        System.out.println("7. Display");
        System.out.println("Enter an Option");

        option= sc.nextInt();
        switch(option){
            case 1:{

                h.directMethod();
                System.out.println("Do you want to continue? ");
                break;
            }
            case 2:{

              h.subtractionMethod();
                System.out.println("Do you want to continue? ");
                break;
            }
            case 3:{

              h.ModuloDivisionMethod();
              System.out.println("do you want to continue? :");
              break;
            }
            case 4:{

              h.ModuloDivisionWithCollision();
                System.out.println("Do you want to continue? ");
                break;
```

```
                }
                case 5:{

                  h.digitExtractionMethod();
                    System.out.println("Do you want to continue? ");
                    break;
                }
                case 6:{

                  h.digitExtractionMethodwithCollision();
                    System.out.println("Do you want to continue? ");
                    break;
                }
                case 7:{

                  h.display();
                    System.out.println("Do you want to continue? ");
                    break;
                }

            }
            ch = sc.next().charAt(0);

        }

        }

}
```

Shivam Khaw
Div – B, Roll No.26

Hash Table

```
package Tree;
import java.util.Scanner;
public class HashTable {
   private final long[] arr = new long[20]; // Fixed-size hash table

   // Constructor
   public HashTable() {
      for (int i = 0; i < arr.length; i++) {
         arr[i] = 0; // Initialize all elements to 0
      }
   }
   // Direct method (hashing based on key value)
   public void direct(Scanner sc) {
      System.out.println("Enter key (1 to 19):");
      int key = sc.nextInt();
      if (key >= 1 && key < 20) {
         arr[key] = key;
      } else {
         System.out.println("Invalid Key! Key must be between 1 and 19.");
      }
   }
   // Subtraction method (hashing using subtraction)
   public void sub(Scanner sc) {
      System.out.println("Enter key (1 to 99):");
      int key = sc.nextInt();
      if (key >= 1 && key <= 99) {
         int addr = 100 - key;
         if (addr < arr.length) {
            arr[addr] = key;
         } else {
            System.out.println("Address exceeds hash table size.");
         }
      } else {
         System.out.println("Invalid Key! Key must be between 1 and 99.");
      }
   }

   // Modulo method with collision handling
   public void moduloWithCollision(Scanner sc) {
      System.out.println("Enter key:");
      int key = sc.nextInt();
      int addr = key % arr.length;
      while (arr[addr] != 0) {
```

```
            addr = (addr + 1) % arr.length; // Handle collision by linear probing
        }
        arr[addr] = key;
    }

    // Display the hash table
    public void display() {
        System.out.println("Hash Table Contents:");
        for (int i = 0; i < arr.length; i++) {
            System.out.printf("Index %d: %d%n", i, arr[i]);
        }
    }

    // Main method
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashTable hashTable = new HashTable(); // Create an object of the HashTable class

        while (true) {
            System.out.println("\nSelect Hashing Method:");
            System.out.println("1. Direct");
            System.out.println("2. Subtraction");
            System.out.println("3. Modulo with Collision Handling");
            System.out.println("4. Display");
            System.out.println("5. Exit");

            int choice = sc.nextInt();

            switch (choice) {
                case 1 -> hashTable.direct(sc);
                case 2 -> hashTable.sub(sc);
                case 3 -> hashTable.moduloWithCollision(sc);
                case 4 -> hashTable.display();
                case 5 -> {
                    System.out.println("Exiting program...");
                    sc.close();
                    return;
                }
                default -> System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```

Shivam Khaw
Div – B, Roll No.26

Adjacency Matrix

Adjacency Matrix :

```
package Graph;

import java.util.Scanner;

public class AM {

        private int [][] adjmatrix = new int[20][20];
        private int [] visitedArray = new int[20];
        private int n;

        public AM() {
        for (int i = 0; i < 20; i++) {
        for (int j = 0; j < 20; j++) {
                adjmatrix[i][j] = 0;

        }
       visitedArray[i] = 0;
        }
        }

        public void createGraph() {
        Scanner scanner = new Scanner(System.in);
        int i, maxEdge, origin, destination;
    System.out.print("Enter Number of vertices: ");
        n = scanner.nextInt();
        maxEdge = (n * (n - 1)) / 2;
    System.out.println("\nEnter the value of edges in adjacency matrix:");
        for (i = 1; i <= maxEdge; i++) {
       System.out.print("Enter 0 0 to exit or enter origin and destination for: " + i + "\n");
       origin = scanner.nextInt();
       destination = scanner.nextInt();
        if ((origin == 0) || (destination == 0)) {
          break;
        }
        if ((origin > n) || (origin < 0) || (destination > n) || (destination < 0)) {
          System.out.println("Invalid inputs");
          i--;
          return;
        } else {
                adjmatrix[origin][destination] = 1;
                adjmatrix[destination][origin] = 1;
        }
        }
        }
```

```
    public void displayMatrix() {
  System.out.println("\nFinal Adjacency Matrix:");
    for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
      System.out.print(adjmatrix[i][j] + " ");
    }
   System.out.println();
    }
    }




    public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            AM d = new AM();
    char ch = 'y';
    int origin, destination, option;

    while(ch == 'y') {
            System.out.println("Selection operation:");
   System.out.println("1. Create Graph");
   System.out.println("2. display Graph");

   System.out.print("Enter an option: ");
   option = scanner.nextInt();
   switch(option) {
    case 1:{
            d.createGraph();
      System.out.print("\nDo you want to continue? (y/n): ");
      break;

    }
    case 2:{
            d.displayMatrix();
      System.out.print("\nDo you want to continue? (y/n): ");
      break;
    }


    }
    ch = scanner.next().charAt(0);

    }
  scanner.close();
    }

}
```

Shivam Khaw
Div – B, Roll No.26

Graph

Undirected Graph :

```
package Graph;
import java.util.Scanner;
class UNG {
        private int[][] adjMatrix = new int[20][20];
        private int n;
        public UNG() {
        for (int i = 0; i < 20; i++) {
        for (int j = 0; j < 20; j++) {
        adjMatrix[i][j] = 0;
        }
          }
        }
        public void createGraph() {
        Scanner scanner = new Scanner(System.in);
        int i, maxEdge, origin, destination;
        System.out.print("Enter Number of vertices: ");
        n = scanner.nextInt();
        maxEdge = (n * (n - 1)) / 2;
        System.out.println("\nEnter the value of edges in adjacency matrix:");
        for (i = 1; i <= maxEdge; i++) {
        System.out.print("Enter 0 0 to exit or enter origin and destination for: " + i + "\n");
        origin = scanner.nextInt();
        destination = scanner.nextInt();
        if ((origin == 0) || (destination == 0)) {
        break;
        }
        if ((origin > n) || (origin < 0) || (destination > n) || (destination < 0)) {
          System.out.println("Invalid inputs");
        i--;
        return;
        } else {
        adjMatrix[origin][destination] = 1;
        adjMatrix[destination][origin] = 1;
        }
        }
        }

        public void insertVertex() {
        n++;
        System.out.println("Number of vertices are: " + n);
        for (int i = 0; i <= n; i++) {
        adjMatrix[i][n] = 0;
        adjMatrix[n][i] = 0;
        }
        }

        public void insertEdge(int origin, int destination) {
```

```
if ((origin > n) || (origin > n)) {
System.out.println("Source or Destination does not exist");
return;
}
adjMatrix[origin][destination] = 1;
adjMatrix[destination][origin] = 1;
}

public void deleteVertex() {
System.out.println("Number of vertices are: " + n);
for (int i = 0; i <= n; i++) {
adjMatrix[i][n] = 0;
adjMatrix[n][i] = 0;
}
n--;
}

public void deleteEdge(int origin, int destination) {
if ((origin > n) || (origin > n)) {
System.out.println("Source or Destination does not exist");
return;
}
adjMatrix[origin][destination] = 0;
adjMatrix[destination][origin] = 0;
}

public void displayMatrix() {
System.out.println("\nFinal Adjacency Matrix:");
for (int i = 1; i <= n; i++) {
for (int j = 1; j <= n; j++) {
System.out.print(adjMatrix[i][j] + " ");
}
System.out.println();
}
}

public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
UNG a = new UNG();
char ch = 'y';
int origin, destination, option;
while (ch == 'y') {
System.out.println("Selection operation:");
System.out.println("1. Create Graph");
System.out.println("2. Insert Vertex");
System.out.println("3. Insert Edge");
System.out.println("4. Delete Vertex");
System.out.println("5. Delete Edge");
System.out.println("6. Display Final Matrix");
System.out.print("Enter an option: ");
option = scanner.nextInt();
```

```
        switch (option) {
        case 1:
                a.createGraph();
                System.out.print("\nDo you want to continue? (y/n): ");
                break;
        case 2:
                a.insertVertex();
                a.displayMatrix();
                System.out.print("\nDo you want to continue? (y/n): ");
                break;
        case 3:
            System.out.print("Enter source & Destination: ");
                origin = scanner.nextInt();
                destination = scanner.nextInt();
                a.insertEdge(origin, destination);
                a.displayMatrix();
                System.out.print("\nDo you want to continue? (y/n): ");
                break;
        case 4:
                a.deleteVertex();
                a.displayMatrix();
                System.out.print("\nDo you want to continue? (y/n): ");
                break;
        case 5:
            System.out.print("Enter source & Destination: ");
                origin = scanner.nextInt();
                destination = scanner.nextInt();
                a.deleteEdge(origin, destination);
                a.displayMatrix();
                System.out.print("\nDo you want to continue? (y/n): ");
                break;
        case 6:
                a.displayMatrix();
                System.out.print("\nDo you want to continue? (y/n): ");
                break;
        }
        ch = scanner.next().charAt(0);
        }
        scanner.close();
        }
}
```

Shivam Khaw
Div – B, Roll No.26

Directed Graph :

```
package Graph;
import java.util.Scanner;
class DG {
        private int[][] adjMatrix = new int[20][20];
        private int n;
        public DG() {
        for (int i = 0; i < 20; i++) {
        for (int j = 0; j < 20; j++) {
        adjMatrix[i][j] = 0;
        }
          }
        }
        public void createGraph() {
        Scanner scanner = new Scanner(System.in);
        int i, maxEdge, origin, destination;
        System.out.print("Enter Number of vertices: ");
        n = scanner.nextInt();
        maxEdge = (n * (n - 1)) / 2;
        System.out.println("\nEnter the value of edges in adjacency matrix:");
        for (i = 1; i <= maxEdge; i++) {
        System.out.print("Enter 0 0 to exit or enter origin and destination for: " + i + "\n");
        origin = scanner.nextInt();
        destination = scanner.nextInt();
        if ((origin == 0) || (destination == 0)) {
        break;
        }
        if ((origin > n) || (origin < 0) || (destination > n) || (destination < 0)) {
          System.out.println("Invalid inputs");
        i--;
        return;
        } else {
        adjMatrix[origin][destination] = 1;

        }
        }
        }

        public void insertVertex() {
        n++;
        System.out.println("Number of vertices are: " + n);
        for (int i = 0; i <= n; i++) {
        adjMatrix[i][n] = 0;
```

```
}
}

public void insertEdge(int origin, int destination) {
if ((origin > n) || (origin > n)) {
System.out.println("Source or Destination does not exist");
return;
}
adjMatrix[origin][destination] = 1;

}

public void deleteVertex() {
System.out.println("Number of vertices are: " + n);
for (int i = 0; i <= n; i++) {
adjMatrix[i][n] = 0;

}
n--;
}

public void deleteEdge(int origin, int destination) {
if ((origin > n) || (origin > n)) {
System.out.println("Source or Destination does not exist");
return;
}
adjMatrix[origin][destination] = 0;

}

public void displayMatrix() {
System.out.println("\nFinal Adjacency Matrix:");
for (int i = 1; i <= n; i++) {
for (int j = 1; j <= n; j++) {
System.out.print(adjMatrix[i][j] + " ");
}
System.out.println();
}
}

public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
DG a = new DG();
```

```
char ch = 'y';
int origin, destination, option;
while (ch == 'y') {
System.out.println("Selection operation:");
System.out.println("1. Create Graph");
System.out.println("2. Insert Vertex");
System.out.println("3. Insert Edge");
System.out.println("4. Delete Vertex");
System.out.println("5. Delete Edge");
System.out.println("6. Display Final Matrix");
System.out.print("Enter an option: ");
option = scanner.nextInt();
switch (option) {
case 1:
        a.createGraph();
        System.out.print("\nDo you want to continue? (y/n): ");
        break;
case 2:
        a.insertVertex();
        a.displayMatrix();
        System.out.print("\nDo you want to continue? (y/n): ");
        break;
case 3:
    System.out.print("Enter source & Destination: ");
        origin = scanner.nextInt();
        destination = scanner.nextInt();
        a.insertEdge(origin, destination);
        a.displayMatrix();
        System.out.print("\nDo you want to continue? (y/n): ");
        break;
case 4:
        a.deleteVertex();
        a.displayMatrix();
        System.out.print("\nDo you want to continue? (y/n): ");
        break;
case 5:
    System.out.print("Enter source & Destination: ");
        origin = scanner.nextInt();
        destination = scanner.nextInt();
        a.deleteEdge(origin, destination);
        a.displayMatrix();
        System.out.print("\nDo you want to continue? (y/n): ");
        break;
case 6:
```

```
                a.displayMatrix();
                System.out.print("\nDo you want to continue? (y/n): ");
                break;
        }
        ch = scanner.next().charAt(0);
        }
        scanner.close();
        }
}
```

Shivam Khaw
Div – B, Roll No.26

Undirected DFS

```
package Graph;
import java.util.Scanner;
class DFS {
   private int[][] adjMatrix = new int[20][20];
   private int[] visitedarr=new  int[20];
   private int n;

   public DFS() {
      for (int i = 0; i < 20; i++) {
         for (int j = 0; j < 20; j++) {
            adjMatrix[i][j] = 0;
         }
         visitedarr[i]=0;

      }

   }
   public void createGraph() {
      Scanner scanner = new Scanner(System.in);
      int i, maxEdge, origin, destination;
      System.out.print("Enter Number of vertices: ");
      n = scanner.nextInt();
      maxEdge = (n * (n - 1)) / 2;
      System.out.println("\nEnter the value of edges in adjacency matrix:");
      for (i = 1; i <= maxEdge; i++) {
         System.out.print("Enter 0 0 to exit or enter origin and destination for: " + i + "\n");
         origin = scanner.nextInt();
         destination = scanner.nextInt();
         if ((origin == 0) || (destination == 0)) {
            break;
         }
         if ((origin > n) || (origin < 0) || (destination > n) || (destination < 0)) {
            System.out.println("Invalid inputs");
            i--;
            return;
         } else {
            adjMatrix[origin][destination] = 1;
            adjMatrix[destination][origin] = 1;
         }
      }
   }
```

```
public void displayMatrix() {
   System.out.println("\nFinal Adjacency Matrix:");
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
         System.out.print(adjMatrix[i][j] + " ");
      }
      System.out.println();
   }
}

void dfs(int x) {

        int j;
        visitedarr[x]=1;
        System.out.println(x + " is visited");
        for ( j = 1; j <=n; j++) {
                        if (adjMatrix[x][j]==1 && visitedarr[j]==0) {
                                dfs(j);
                        }
                }
}
public static void main(String[] args) {
   Scanner scanner = new Scanner(System.in);
   DFS a = new DFS();
   char ch = 'y';
   int origin, destination, option;
   while (ch == 'y') {
      System.out.println("Selection operation:");
      System.out.println("1. Create Graph");
      System.out.println("2. DFS");
      System.out.println("3. Display Final Matrix");
      System.out.print("Enter an option: ");
      option = scanner.nextInt();
      switch (option) {
         case 1:
            a.createGraph();
            System.out.print("\nDo you want to continue? (y/n): ");
            break;

         case 2:
                System.out.println("Enter first verstise::-");
                int x=scanner.nextInt();
                a.dfs(x);
            System.out.print("\nDo you want to continue? (y/n): ");
```

```
            break;

        case 3:
            a.displayMatrix();
            System.out.print("\nDo you want to continue? (y/n): ");
            break;
      }
      ch = scanner.next().charAt(0);
   }
   scanner.close();
}
```

Shivam Khaw
Div – B, Roll No.26

Undirected Graph BFS :

```
package Ugbfs;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

class BFS {
   private int[][] adjMatrix = new int[20][20];
   private int[] visitedarr = new int[20];
   private int n;

   public BFS() {
      for (int i = 0; i < 20; i++) {
         for (int j = 0; j < 20; j++) {
            adjMatrix[i][j] = 0;
         }
         visitedarr[i] = 0;
      }
   }

   public void createGraph() {
      Scanner scanner = new Scanner(System.in);
      int i, maxEdge, origin, destination;
      System.out.print("Enter Number of vertices: ");
      n = scanner.nextInt();
      maxEdge = (n * (n - 1)) / 2;
      System.out.println("\nEnter the value of edges in adjacency matrix:");
      for (i = 1; i <= maxEdge; i++) {
         System.out.print("Enter 0 0 to exit or enter origin and destination for: " + i + "\n");
         origin = scanner.nextInt();
         destination = scanner.nextInt();
         if ((origin == 0) || (destination == 0)) {
            break;
         }
         if ((origin > n) || (origin < 0) || (destination > n) || (destination < 0)) {
            System.out.println("Invalid inputs");
            i--;
            continue; // Allow user to enter inputs again
         } else {
            adjMatrix[origin][destination] = 1;
            adjMatrix[destination][origin] = 1;
         }
      }
   }
```

```
public void displayMatrix() {
    System.out.println("\nFinal Adjacency Matrix:");
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            System.out.print(adjMatrix[i][j] + " ");
        }
        System.out.println();
    }
}

public void bfs(int startVertex) {
    Queue<Integer> queue = new LinkedList<>();
    queue.add(startVertex);
    visitedarr[startVertex] = 1;

    System.out.println("\nBFS Traversal starting from vertex " + startVertex + ":");

    while (!queue.isEmpty()) {
        int vertex = queue.poll();
        System.out.println(vertex + " is visited");

        for (int j = 1; j <= n; j++) {
            if (adjMatrix[vertex][j] == 1 && visitedarr[j] == 0) {
                queue.add(j);
                visitedarr[j] = 1; // Mark node as visited when it is enqueued
            }
        }
    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    BFS bfsGraph = new BFS();
    char ch = 'y';
    int option;

    while (ch == 'y') {
        System.out.println("\nSelection operation:");
        System.out.println("1. Create Graph");
        System.out.println("2. BFS");
        System.out.println("3. Display Final Matrix");
        System.out.print("Enter an option: ");
        option = scanner.nextInt();
```

```
            switch (option) {
                case 1:
                    bfsGraph.createGraph();
                    break;

                case 2:
                    System.out.println("Enter starting vertex:");
                    int startVertex = scanner.nextInt();
                    for (int i = 0; i < 20; i++) {
                        bfsGraph.visitedarr[i] = 0; // Reset visited array before BFS
                    }
                    bfsGraph.bfs(startVertex);
                    break;

                case 3:
                    bfsGraph.displayMatrix();
                    break;

                default:
                    System.out.println("Invalid option. Please select 1, 2, or 3.");
            }

            System.out.print("\nDo you want to continue? (y/n): ");
            ch = scanner.next().charAt(0);
        }

        scanner.close();
    }
}
```

Shivam Khaw
Div – B, Roll No.26

Notations

Infix to postfix

```
package Tree;
import java.util.Stack;
public class infixtopostfix {

        static int Prec(char ch)
        {
            switch (ch)
            {
                case '+':
                case '-':
                    return 1;

                case '*':
                case '/':
                    return 2;

                case '^':
                    return 3;
            }
            return -1;
        }
        // The main method that converts given infix expression to postfix expression.
        static String infixToPostfix(String exp)
        {
            // initializing empty String for result
            String result = new String("");
            // initializing empty stack
            Stack<Character> stack = new Stack<>();

            for (int i = 0; i<exp.length(); ++i)
            {
                char c = exp.charAt(i);

                // If the scanned character is an operand, add it to output.
                if (Character.isLetterOrDigit(c))
                    result += c;

                    // If the scanned character is an '(', push it to the stack.
                else if (c == '(')
                    stack.push(c);

                    //  If the scanned character is an ')', pop and output from the stack
                    // until an '(' is encountered.
                else if (c == ')')
                {
                    while (!stack.isEmpty() && stack.peek() != '(')
                        result += stack.pop();
```

```
            if (!stack.isEmpty() && stack.peek() != '(')
                return "Invalid Expression"; // invalid expression
            else
                stack.pop();
        }
        else // an operator is encountered
        {
            while (!stack.isEmpty() && Prec(c) <= Prec(stack.peek()))
                result += stack.pop();
            stack.push(c);
        }

    }

    // pop all the operators from the stack
    while (!stack.isEmpty())
        result += stack.pop();

    return result;
  }

  // Driver method
  public static void main(String[] args)
  {
    String exp = "a+b*(c^d-e)^(f+g*h)-i";
    System.out.println(infixToPostfix(exp));
  }
}
```

Shivam Khaw
Div – B, Roll No.26

Prims

```
package prims;
import java.util.Arrays;
class Prim {
  public void Pgraph(int G[][], int V) {
    int INF = 999;
    int no_edge; // number of edge
    boolean[] selected = new boolean[V];
    // set selected false initially
    Arrays.fill(selected,  false);
    no_edge = 0;
    selected[0] = true;
  System.out.println("Edge : Weight");
  while (no_edge < V - 1) {
      int min = INF;
      int x = 0; // row number
      int y = 0; // col number
      for (int i = 0; i < V; i++) {
      if (selected[i] == true) {
          for (int j = 0; j < V; j++) {
            // not in selected and there is an edge
            if (!selected[j] && G[i][j] != 0) {
              if (min > G[i][j]) {
                min = G[i][j];
                x = i;
                y = j;
              }
            }
          }
        }
      }
      System.out.println(x + " - " + y + " : " + G[x][y]);
      selected[y] = true;
      no_edge++;
    }
  }
  public static void main(String[] args) {
    Prim g = new Prim();
    int V = 5;
    int[][] G = { { 0, 9, 70, 0, 0 }, { 9, 0, 90, 20, 40 }, { 70, 90, 0, 50, 60 }, { 0, 20, 50, 0, 30 },
        { 0, 40, 60, 30, 0 } };

    g.Pgraph(G, V);
  }
}
```