```
//Problem 1


public class Problem1 {
	public static void main(String[] args) {
		Queue instrqueue = new Queue(4);
		instrqueue.enqueue("Naveen:3");
		instrqueue.enqueue("Raju:2");
		instrqueue.enqueue("Shanu:2");
		instrqueue.enqueue("Radha:1");
		System.out.println("Before changing based on the rank ");
		instrqueue.display();
		Queue outstrqueue = displayBasedOnRank(instrqueue);
	   System.out.println("After changing based on the rank ");
		outstrqueue.display();
	}

	public static Queue displayBasedOnRank(Queue instrqueue) {
		Queue result = new Queue(4);

		String[] arrvalues = new String[instrqueue.getMaxSize()];
		int[] rankarray = new int[instrqueue.getMaxSize()];

		for (int i = 0; i < instrqueue.getMaxSize(); i++) {
			String name = instrqueue.dequeue();
			arrvalues[i] = name;
			int ranklast = Integer.parseInt("" + name.charAt(name.length() -
1));
			rankarray[i] = ranklast;
		}
		Arrays.sort(rankarray);

		for (int i : rankarray) {
			String name= "";
			for (int j = 0; j <=arrvalues.length-1; j++) {
				 name =  arrvalues[j] ;

				char ch = name.charAt(name.length()-1) ;
				int n = Integer.parseInt(ch+ "");
				//System.out.println(n);
				if(i == n )
				{
					result.enqueue(arrvalues[j]);
					break;
				}
			}
		}
		return result;

	}

}




// Problem 2


public class Problem2 {
	public static void main(String[] args) {
		Stack values = new Stack(5);
```

```
		values.push(81);
		values.push(29);
		values.push(12);
		values.push(56);
		values.push(34);
		System.out.println("Before");
		values.display();
		Stack result = replaceSTack(values);
		System.out.println();
		System.out.println("After");
		result.display();

	}

	public static Stack replaceSTack(Stack data) {

		Stack values = new Stack(data.getMaxValue());
		Stack result = new Stack(data.getMaxValue());
		Stack odd = new Stack(data.getMaxValue());
		Stack even = new Stack(data.getMaxValue());
		while (!data.isEmpty()) {
			int n = data.pop();
			int sum = sumOfDigits(n);
			if (sum % 2 == 0) {
				even.push(n);

			} else {
				odd.push(n);
			}
		}

		while (!odd.isEmpty()) {
			int n = odd.pop();
			values.push(n);
		}
		while (!even.isEmpty()) {
			int n = even.pop();
			values.push(n);
		}
		while (!values.isEmpty()) {
			int n = values.pop();
			result.push(n);
		}


		return result;

	}
}



//Problem3


public class Problem3 {
	public static void main(String[] args) {
		Queue intstrqueue1 = new Queue(10);
		intstrqueue1.enqueue("Laptop:Electronics");
```

```
		intstrqueue1.enqueue("Apple:Fruit");
		intstrqueue1.enqueue("Shampoo:Toilery");
		Queue intstrqueue2 = new Queue(10);
		intstrqueue2.enqueue("tablet:Electronics");
		intstrqueue2.enqueue("Banana:Fruit");
		intstrqueue2.enqueue("Soap:Toilery");
		Queue outputqueue = matchCotnet(intstrqueue1, intstrqueue2);
		outputqueue.display();

	}

	public static Queue matchCotnet(Queue intstrqueue1 , Queue intstrqueue2) {
		Queue result = new Queue(10);

		while( !intstrqueue1.isEmpty()) {
			String s= intstrqueue1.dequeue();
			String name1=dataString(s);

			String s2= intstrqueue2.dequeue();
			String name2=dataString(s2);

			String res= name1 +"-"+ name2 ;
			result.enqueue(res);


		}
		return result;

	}
	public static String  dataString(String s ) {
		String name = "";
		for (int i = 0; i <=s.length()-1; i++) {
			String ch = s.charAt(i)+"";
			if(s.charAt(i) == ':') {
				break;
			}
			else {
				name =name.concat(ch+"");
			}
		}

		return name;

	}

}



//Problem 4
for this question output they given and my output is different

public class Problem4 {

	public static void main(String[] args) {
		Stack values = new Stack(5);
		values.push(12);
		values.push(15);
		values.push(9);
		values.push(12);
		values.push(21);
		Stack outputStack = primeFactors(values);
		 outputStack.display();
```

```
	}

	public static Stack primeFactors(Stack stack) {
		Stack temp1 = new Stack(5);
		Stack temp2 = new Stack(5);
		Stack result = new Stack(5);
		while (!stack.isEmpty()) {
			int n = stack.pop();
		int count = 0;
			for (int i = 1; i <=n ; i++) {

				if (n % i == 0) {
					int countprime =0;
					for (int j = 1; j <=i; j++) {
						if (i % j == 0) {
							countprime++;
						}
					}
					if (countprime == 2) {
						count++;
						//System.out.println(i +" " + n + " "+
count);

					}
				}

			}
			//System.out.println(count + " "+n);
			if (count % 2 == 0) {
			//	System.out.print(n);
				temp2.push(n);
			} else {
				temp1.push(n);
			}

		}
		//temp1.display();
		while (!temp1.isEmpty()) {
			result.push(temp1.pop());
		}
		while (!temp2.isEmpty()) {
			result.push(temp2.pop());
		}
		return result;

	}

}


//problem 5


public class Problem5 {
	public static void main(String[] args) {

		Queue values = new Queue(5);

		values.enqueue(2);
		values.enqueue(7);
		values.enqueue(5);
		values.enqueue(10);
```

```
		Queue values2 = new Queue(5);

		values2.enqueue(11);
		values2.enqueue(1);
		values2.enqueue(8);

		Queue resQueue = sumofprimePairs(values, values2);
		resQueue.display();

	}

	public static Queue sumofprimePairs(Queue que1, Queue que2) {
		Queue result = new Queue(que1.getMaxSize() + que2.getMaxSize());

		while (!que1.isEmpty()) {
			int n1 = que1.dequeue();
			if(! que2.isEmpty()) {
				int n2 = que2.dequeue();
				int sum = n1 + n2;
				int count = 0;
				for (int i = 2; i <= sum / 2; i++) {
					if (sum % i == 0) {
						count++;
					}
				}
				if (count == 0) {
					result.enqueue(n1);
					result.enqueue(n2);
				}
			}
		}

		return result;

	}

}


//problem 6

public class Problem6 {
	public static void main(String[] args) {
		Stack values = new Stack(5);
		values.push(4);
		values.push(8);
		values.push(9);
		//values.push(10);

		Queue values2 = new Queue(5);

		values2.enqueue(2);
		values2.enqueue(4);
		values2.enqueue(16);
		//values2.enqueue(16);

		Queue resQueue = sumofprimePairs(values, values2);
		//resQueue.display();
	}
	public static Queue sumofprimePairs(Stack stack, Queue que2) {
		Queue result = new Queue(stack.getMaxValue() + que2.getMaxSize());

		while (!stack.isEmpty()) {
```

```
                        int n1 = stack.pop();
                        int resvaluestack = n1 * n1;
                        if(! que2.isEmpty()) {
                                int n2 = que2.dequeue();
                                int resvaluequeue = n2*n2*n2;
                                if(resvaluestack == resvaluequeue) {
                                        result.enqueue(n1);
                                        result.enqueue(n2);
                                }

                        }
                }
                while (!result.isEmpty()) {
                        System.out.println(result.dequeue());

                }
                return result;

        }

}



//problem 7

////7th problem
public class ExamExample {
        public static void main(String[] args) {
                ExamExample [] classNames = new ExamExample[5];
                Stack intstrstack = new Stack(10);

                intstrstack.push("b");
                intstrstack.push("Z");
                intstrstack.push("5");

                Queue intstrqueue = new Queue(10);
                intstrqueue.enqueue("ab5c");
                intstrqueue.enqueue("2");
                intstrqueue.enqueue("Qwd4zs");
                intstrqueue.enqueue("4");
                intstrqueue.enqueue("bige");
                intstrqueue.enqueue("0");
                Queue result = resultData(intstrstack, intstrqueue);
                result.display();

        }
        public static Queue resultData(Stack stack , Queue queue) {

                Queue valuesResult = new Queue(queue.getMaxSize()) ;

                while ( !stack.isEmpty()) {
                        String  s= stack.pop();
                        String name = queue.dequeue();
                        int num=Integer.parseInt(queue.dequeue()) ;
                        String s1 = name.charAt(num)+ "";
                        if(s.equals(s1)) {
                                valuesResult.enqueue(s);
                        }else {
                                valuesResult.enqueue("-1");
                        }
                }
                return valuesResult;
```

```
	}
}
```

problem 8

----->Problem 8 need to do

problem 10

```
public class Problem10 {
	public static void main(String[] args) {

		Queue values = new Queue(10);

		values.enqueue(2);
		values.enqueue(7);
		values.enqueue(4);
		values.enqueue(9);
		values.enqueue(5);
		values.enqueue(2);
		values.enqueue(10);

		Queue values2 = new Queue(10);

		values2.enqueue(3);
		values2.enqueue(6);
//		values2.enqueue(5);
//	values2.enqueue(2);
//		values2.enqueue(10);

		Queue resQueue = sumofprimePairs(values, values2);
		resQueue.display();
	}
	public static Queue sumofprimePairs(Queue que1, Queue que2)  {
		Queue result = new Queue(que1.getMaxSize() + que2.getMaxSize());
		while(!que1.isEmpty() || !que2.isEmpty()) {
			int n =0;
			int n1 =0;
			if(!que1.isEmpty()) {
				 n = que1.dequeue();
				// System.out.println(n);
			}
			if(!que2.isEmpty()) {
				 n1 = que2.dequeue();
			}

			if(n !=0 && n1 !=0) {
				if(n %2 == 0) {
					result.enqueue(n+n1);
				}
				else {
					result.enqueue( n-n1);
				}
			}
			else {
				int val = n > n1 ? n : n1;
				result.enqueue(val);
			}
		}
		return result;
```

```
	}

}



//problem 11

public class Problem11 {
	public static void main(String[] args) {
		//Queue intstrqueue1 = new Queue(10);
		Stack intstrstack = new Stack(10);

		intstrstack.push("Alice:D1");
		intstrstack.push("Bob02");
		intstrstack.push("Eve:D3");
		intstrstack.push("Oscar:D1");
		intstrstack.push("Charlie02");
		Queue outqueue = vowelArrange(intstrstack);

	}
	public static Queue vowelArrange(Stack intstrstack ) {
		Queue temp1 = new Queue(10);
		Queue temp2 = new Queue(10);
		Queue result = new Queue(10);

		while(! intstrstack.isEmpty()) {
			String name = intstrstack.pop();
			String ch = name.charAt(0)+"";
			if(ch.equalsIgnoreCase("a") || ch.equalsIgnoreCase("e") ||
ch.equalsIgnoreCase("i") ||
					ch.equalsIgnoreCase("o") ||
ch.equalsIgnoreCase("u")) {
				temp1.enqueue(name);

			}
			else {
				temp2.enqueue(name);
			}
		}

		while(! temp1.isEmpty()) {
			String name = temp1.dequeue();
			System.out.println(name);
			result.enqueue(name);

		}
		while(! temp2.isEmpty()) {
			String name = temp2.dequeue();
			result.enqueue(name);

		}
		result.display();
		return null;

	}

}


// problem 12
```

```
public class Poblem12 {
	public static void main(String[] args) {
		//Queue intstrqueue1 = new Queue(10);
		Stack intstrstack = new Stack(10);

		intstrstack.push("Laptop-A");
		intstrstack.push("Chair-B");
		intstrstack.push("Book-C");
		intstrstack.push("Phone-A");
		intstrstack.push("Table-B");
		Queue outqueue = vowelArrange(intstrstack);
		outqueue.display();

	}
	public static Queue vowelArrange(Stack intstrstack ) {
		Queue temp1 = new Queue(10);
		Queue temp2 = new Queue(10);
		Queue temp3 = new Queue(10);
		Queue result = new Queue(10);

		while(! intstrstack.isEmpty()) {
			String name = intstrstack.pop();
			String ch = name.charAt(name.length()-1)+"";
			if(ch.equals("A")) {
				temp1.enqueue(name);
			}
			else if(ch.equals("B")) {
				temp2.enqueue(name);
			}
			else if(ch.equals("C")) {
				temp3.enqueue(name);
			}

		}

		while(! temp1.isEmpty()) {
			String name = temp1.dequeue();
			result.enqueue(name);

		}
		while(! temp2.isEmpty()) {
			String name = temp2.dequeue();
			result.enqueue(name);

		}
		while(! temp3.isEmpty()) {
			String name = temp3.dequeue();
			result.enqueue(name);

		}

		return result;

	}

}
```