/\* Q1 A

PROGRAM TO PRINT

1

12

123

1234

12345

\*/

public class Pattern {

public static void main(String args[]) {

int i, j;

for (i = 1; i <= 5; i++) {

for (j = 1; j <= i; j++) {

System.out.print(j);

}

System.out.println();

}

}

}

/\* Q1 B

PROGRAM TO PRINT

54321

5432

543

54

5

\*/

public class Pattern {

public static void main(String args[]) {

int i, j;

for (i = 1; i <= 5; i++) {

for (j = 5; j >= i; j--) {

System.out.print(j);

}

System.out.println();

}

}

}

/\* Q1 C

PROGRAM TO PRINT

x

xxx

xxxxx

xxxxxxx

xxxxx

xxx

x

\*/

public class Pattern {

public static void main(String[] args) {

int n = 4;

char c = 'X';

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n - i; j++) {

System.out.print(" ");

}

for (int j = 1; j <= i \* 2 - 1; j++) {

System.out.print(c);

}

System.out.println();

}

for (int i = n - 1; i > 0; i--) {

for (int j = 1; j <= n - i; j++) {

System.out.print(" ");

}

for (int j = 1; j <= i \* 2 - 1; j++) {

System.out.print(c);

}

System.out.println();

}

}

}

// Q2 Program to take the input and determine if it is prime number or not.

import java.util.Scanner;

public class Prime {

public static void main(String[] args) {

int num, i, count = 0;

Scanner scan = new Scanner(System.in);

System.out.print("Enter a Number : ");

num = scan.nextInt();

for (i = 2; i < num; i++) {

if (num % i == 0) {

count++;

break;

}

}

if (count == 0) {

System.out.print("This is a Prime Number");

} else {

System.out.print("This is not a Prime Number");

}

}

}

// Q3 Program to display the fibonacci series till less than 200 using only 2 variables.

public class Series {

public static void main(String[] args) {

System.out.println("0");

System.out.println("1");

int a = 0;

int b = 1;

for (int i = 1; i < 12; i++) {

int nextNumber = a + b;

System.out.println(nextNumber);

a = b;

b = nextNumber;

}

}

}

// Q5 Program to check if a name is palindrome.

import java.util.Scanner;

public class Palindrome {

public static void main(String[] args) {

String str, rev = "";

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string:");

str = sc.nextLine();

int length = str.length();

for (int i = length - 1; i >= 0; i--) {

rev = rev + str.charAt(i);

}

if (str.equals(rev)) {

System.out.println(str + " is a palindrome");

} else {

System.out.println(str + " is not a palindrome");

}

}

}

// Q6 Program to check armstrong.

import java.util.Scanner;

public class armstrong {

public static void main(String[] args) {

int c = 0, a, n, temp;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number:");

n = sc.nextInt();

temp = n;

while (n > 0) {

a = n % 10;

n = n / 10;

c = c + (a \* a \* a);

}

if (temp == c) {

System.out.println("Armstrong number");

} else {

System.out.println("Not armstrong number");

}

}

}

// Q7 Program to check Factorial.

import java.util.Scanner;

public class Factorial {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number:");

int n = sc.nextInt();

int result = 1;

for (int i = n; i >= 1; i--) {

result = result \* i;

}

System.out.println("The factorial of " + n + " is " + result);

}

}

// Q8 Program to check wovel.

import java.util.Scanner;

public class Check {

public static void main(String[] args) {

int i = 0;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a character : ");

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

if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' || ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {

System.out.println(ch + " is Vowel");

} else if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z')) {

System.out.println(ch + " is Consonant");

} else {

System.out.println("Not an alphabet");

}

}

}

// Q9 Simple Calculator Program.

import java.util.Scanner;

class Calculator

{

public static void main(String[] args)

{

Scanner s=new Scanner(System.in);

System.out.println("YOU HAVE FOLLOWING CHOICES : ");

System.out.println("1. ADDITION");

System.out.println("2. SUBTRACTION ");

System.out.println("3. MULTIPLICATION ");

System.out.println("4. DIVISION");

System.out.println("ENTER YOUR CHOICE : ");

int i=s.nextInt();

System.out.println("ENTER FIRST NUMBER ");

int a=s.nextInt();

System.out.println("ENTER SECOND NUMBER ");

int b=s.nextInt();

double result=0;//'result' will store the result of operation

switch(i)

{

case 1: result=a+b; break;

case 2: result=a-b; break;

case 3: result=a\*b; break;

case 4:

if(b==0)//when denominator becomes zero

{

System.out.println("DIVISION NOT POSSIBLE"); break;

}

else

result=a/b;

default:

System.out.println("YOU HAVE ENTERED A WRONG CHOICE");

}

System.out.println("RESULT = "+result);

}

}

// Q10 Program to copy one array into another

public class Copy {

public static void main(String[] args) {

int a[] = {10, 20, 30, 40, 50};

int b[] = new int[a.length];

for (int i = 0; i < a.length; ++i) {

b[i] = a[i];

}

System.out.println("Array 1: ");

for (int i = 0; i < a.length; ++i) {

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

}

System.out.println("\nArray 2: ");

for (int i = 0; i < b.length; ++i) {

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

}

}

}

Q12.

import java.lang.\*;

import java.io.\*;

import java.util.\*;

class ReverseString

{

public static void main(String[] args)

{

String input = "GeeksforGeeks";

byte [] strAsByteArray = input.getBytes();

byte [] result =

new byte [strAsByteArray.length];

for (int i = 0; i<strAsByteArray.length; i++)

result[i] =

strAsByteArray[strAsByteArray.length-i-1];

System.out.println(new String(result));

}

}

Q13. Checked- Checked exception are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword.

Unchecked- Unchecked Exception are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions.

In Java exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.

Q14-

Throw- Throw keyword is used to throw an exception explicitly. If we see syntax wise than throw is followed by an instance of Exception class. Throw keyword is used in the method body to throw an exception.

For Example:

void myMethod() {

try {

throw new ArithmeticException("Something went wrong!!");

}

catch (Exception exp) {

System.out.println("Error: "+exp.getMessage());

}

}

Throws- Throws clause is used to declare an exception, which means it works similar to the try-catch block. If we see syntax wise than throws is followed by exception class names. Throws is used in method signature to declare the exceptions that can occur in the statements present in the method.

For Example

void sample() throws ArithmeticException{

}

Q15-

When a try catch block is present in another try block then it is called the nested try catch block. Each time a try block does not have a catch handler for a particular exception, then the catch blocks of parent try block are inspected for that exception, if match is found that that catch block executes.If neither catch block nor parent catch block handles exception then the system generated message would be shown for the exception, similar to what we see when we don’t handle exception.

Code:

try {

statement 1;

statement 2;

try {

statement 3;

statement 4;

try {

statement 5;

statement 6;

}

catch(Exception e2) {

}

}

catch(Exception e1) {

}

}

catch(Exception e3) {

}

Q16.

Multitasking - Multitasking has the same meaning of multiprogramming but in a more general sense, as it refers to having multiple (programs, processes, tasks, threads) running at the same time. This term is used in modern operating systems when multiple tasks share a common processing resource (e.g., CPU and Memory). At any time the CPU is executing one task only while other tasks waiting their turn. The illusion of parallelism is achieved when the CPU is reassigned to another task (i.e. process or thread context switching).

Multithreading - Multithreading is an execution model that allows a single process to have multiple code segments (i.e., threads) run concurrently within the “context” of that process. You can think of threads as child processes that share the parent process resources but execute independently. Multiple threads of a single process can share the CPU in a single CPU system or (purely) run in parallel in a multiprocessing system

Q17.

The java.util.Deque interface is a subtype of the java.util.Queue interface. The Deque is related to the double-ended queue that supports addition or removal of elements from either end of the data structure.

Code:

public class Deque2

{

public static void main( String[] args )

{

Deque deque = new Deque();

deque.add( "ape" );

deque.add( "bat" );

deque.add( "cat" );

deque.add( "bat" );

deque.add( "bat" );

deque.add( "cat" );

System.out.println( deque );

System.out.println();

System.out.println( "deque.count( bat ) = " + deque.count( "bat" ) );

int index = deque.indexOf( "bat" );

System.out.println( "deque.indexOf( bat ) = " + index );

deque.remove( index );

System.out.println( "After deque.remove( " + index + " ) = " + deque );

deque.replace( 0, 2, "bat", "BAT" );

System.out.println( "After deque.replace( 0, 2, bat, BAT ) = " + deque );

System.out.println( "deque.remove( cat ) = " + deque.remove( "cat" ) );

System.out.println( "After deque.remove( cat ) = " + deque );

deque.remove( deque.begin() );

System.out.println( "After deque.remove( begin() ) = " + deque );

}

}

Q18.

Q19.MAP INTERFACE

The Map interface represents a mapping between a key and a value. The Map interface is not a subtype of the Collection interface. Therefore it behaves a bit different from the rest of the collection types. A Map cannot contain duplicate keys and each key can map to at most one value. Some implementations allow null key and null value (HashMap and LinkedHashMap) but some do not (TreeMap).

The order of a map depends on specific implementations, e.g TreeMap and LinkedHashMap have predictable order, while HashMap does not.

Q20.

Arrya List:

1) ArrayList internally uses dynamic array to store the elements.

2) Manipulation with ArrayList is slow because it internally uses array. If any element is removed from the array, all the bits are shifted in memory.

3) ArrayList class can act as a list only because it implements List only.

4) ArrayList is better for storing and accessing data.

LINKED LIST:

1) LinkedList internally uses doubly linked list to store the elements.

2) Manipulation with LinkedList is faster than ArrayList because it uses doubly linked list so no bit shifting is required in memory.

3) LinkedList class can act as a list and queue both because it implements List and Deque interfaces.

4) LinkedList is better for manipulating data.

Q21 The ArrayList class extends AbstractList and implements the List interface. ArrayList supports dynamic arrays that can grow as needed. Standard Java arrays are of a fixed length. After arrays are created, they cannot grow or shrink, which means that you must know in advance how many elements an array will hold. Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.

Q22 System class provides the following functionalities:

1. Standard output
2. Error output streams
3. Standard input and access to externally defined properties and environment variables.
4. A utility method for quickly copying a portion of an array.
5. A means of loading files and libraries

Q23. OutputStreamWrite

Q24. FileReader

Q25. InputStreamReader

Q26. **Java FileInputStream-** Java FileInputStream class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc. You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

**Java FileOutputStream** -Java FileOutputStream is an output stream used for writing data to a file.If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

**Java RandomAccessFile -**Java RandomAccessFile provides facility to both read and write data to a file. RandomAccessFile works with file as large array of bytes stored in the file system and a cursor using which we can move the file pointer position.

Q28.

System.in is used to take input from command prompt and System.out is used for output at command prompt. Every Operating System comes with three types builtin streams used by OS. A stream is nothing but a carrier of data from one place to another. The streams are:

1. in: in stream carries data from Keyboard to CPU. Any data you type at Keyboard like notepad is carried to CPU by this stream. Any language would like to read from Keyboard should connect to in stream. Java connects through System class as System.in. That is System.in is implicitly connected to input mechanism of underlying OS.

2. out: out stream of OS carries data from CPU to command prompt. Any language for output should connect to this stream like Java connects as System.out.

3. err: err stream is also same purpose of carrying data from CPU to Monitor but used by OS for displaying error messages while Operating System is being booted. Of course, Java Programmer can also use this like out stream like System.err.

Q29.

1. System class is provided with useful fields (static members) pertaining to the environment.

2. Standard input,output and error output streams are provided with System class. These are used to access the externally defined properties and environment variables.

3. The System class can not be instantiated.

4. The System class contains several useful class fields and methods.

5. The System class cannot be instantiated.

Q30.

Ans. OutputStreamWriter

Q31 FileReader

Q32 InputStreamReader

Q33 RandomAccessFile treats the file as an array of bytes where it has the internal pointer. The fact that it treats it like a large array of bytes is what is unique about this class. FileInputStream however just reads the stream and returns the data. It is more suited to reading raw data like images etc. It does not treat the file as a large array, it just keeps tabs of where in the file it has read so far. With FileInputStream you would actually have to read the data and place it into an array to get the same style of access as RandomAccessFile.

Q34 A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position: A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes. A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity. A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

Channels class defines static methods that support the interoperation of the stream classes of the java.io package with the channel classes of this package.

Example Code:

RandomAccessFile aFile = new RandomAccessFile("data/nio-data.txt", "rw");

FileChannel inChannel = aFile.getChannel();

ByteBuffer buf = ByteBuffer.allocate(48);

int bytesRead = inChannel.read(buf);

while (bytesRead != -1) {

buf.flip();

while(buf.hasRemaining()){

System.out.print((char) buf.get()); // read 1 byte at a time

}

buf.clear();

bytesRead = inChannel.read(buf);

}

aFile.close();

Q35 The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.

The metadata means data about data i.e. we can get further information from the data. If you have to get metadata of a table like total number of column, column name, column type etc. , ResultSetMetaData interface is useful because it provides methods to get metadata from the ResultSet object.

Q36 DDL (Data Definition Language) allows to add, modify and delete the logical structures which contain the data or which allow users to access or maintain the data.

SNIPPET: CREATE TABLE STUDENTS (ID INT, Name VARCHAR);

DML (Data Manipulation Language) allows to add, modify and delete the data itself.

SNIPPET: INSERT INTO STUDENTS VALUES (1, ‘SAGAR’);

DQL (Data Query Language) is used to retrieve data from the database. SELECT is the main DQL instruction which is used to retrieve data.

SNIPPET: SELECT \* FROM STUDENTS;

DCL (Data Control Language) is used to grant or revoke permissions on databases and their contents.

SNIPPET: GRANT CREATE TABLE TO USER1;

Q37 Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Cascading Style Sheet (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media.

JavaScript (JS), is a high-level, dynamic, weakly typed, prototype-based, multi-paradigm, and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make webpages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins by means of a built-in JavaScript engine.

Q40.

System.in is used to take input from command prompt and System.out is used for output at command prompt. Every Operating System comes with three types builtin streams used by OS. A stream is nothing but a carrier of data from one place to another. The streams are:

1. in: in stream carries data from Keyboard to CPU. Any data you type at Keyboard like notepad is carried to CPU by this stream. Any language would like to read from Keyboard should connect to in stream. Java connects through System class as System.in. That is System.in is implicitly connected to input mechanism of underlying OS.

2. out: out stream of OS carries data from CPU to command prompt. Any language for output should connect to this stream like Java connects as System.out.

3. err: err stream is also same purpose of carrying data from CPU to Monitor but used by OS for displaying error messages while Operating System is being booted. Of course, Java Programmer can also use this like out stream like System.err.

Q41.

The purpose of the System class is to provide access to system resources. It contains accessibility to standard input, standart output, error output streams, current time in millis, terminating the application, etc.

Q42.

Ans. OutputStreamWriter

Q43.

Ans. FileReader

Q44.

Ans.File Output Stream

Q45. **Java FileInputStream-** Java FileInputStream class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc. You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

**Java FileOutputStream** -Java FileOutputStream is an output stream used for writing data to a file.If you have to write primitive values into a file, use FileOutputStream class. You can write byte-oriented as well as character-oriented data through FileOutputStream class. But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream.

**Java RandomAccessFile -**Java RandomAccessFile provides facility to both read and write data to a file. RandomAccessFile works with file as large array of bytes stored in the file system and a cursor using which we can move the file pointer position.

Q46.

A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position: A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes. A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity. A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

Channels class defines static methods that support the interoperation of the stream classes of the java.io package with the channel classes of this package.

Example Code:

RandomAccessFile aFile = new RandomAccessFile("data/nio-data.txt", "rw");

FileChannel inChannel = aFile.getChannel();

ByteBuffer buf = ByteBuffer.allocate(48);

int bytesRead = inChannel.read(buf);

while (bytesRead != -1) {

buf.flip();

while(buf.hasRemaining()){

System.out.print((char) buf.get()); // read 1 byte at a time

}

buf.clear();

bytesRead = inChannel.read(buf);

}

aFile.close();