THINK BRIDGE ASSESSMENT

Q1.After 7 years,will the candidate be a developer or manager?why?

Answer: According to me a candidate with seven years of experience becomes a manager

Becoming a manager is an important step towards becoming a renowed proffessional and climbing the ladder leading to upper management.

In order to become a manager, in the majority of occasions you will need to prove your quality as an employee first. Apart from that, the need for managerial knowledge apart from experience is also vital. It depends on the subject of each work position, your competition and many more relevant factors.

Above 5 years of relevant experience as an employee should make you a strong candidate for a managerial position in field of work.

Managers need to be highly organized and has to show the strength,determination,support for their staff and they need to develop the hard and soft skills necessary to be a leader over time.

Q2.Favorite Subject in college/Learning?

Answer:
My favourite subject in school is English.
It is the common language.if we visit some other countries like japan,France english is helpful for us to communicate with other people.

It is quite easy and pleasant to study it. English is used in each field of life and it is useful to know it.we can watch english movies if we are good at english.With the help of subtitles also we can undertsand the movie even the movie is in tamil,kannada..etc

Most of web sides are in English. It is nice to be able to use such a sides like yahoo, msn and english Wikipedia. I can get a lot of information and news, which are not available on polish sides.

I can use english books and magazines. It is effectively way to study English in home.

Q3.write a coding program on converting currency with decimals into words

Program:

```
import java.text.NumberFormat;

import java.text.ParseException;

import java.util.Locale;import java.util.regex.Pattern;

public final class EnglishNumberTranslator{

        private static final double MAX_LIMIT = 999999.99;

        private static final String OUT_OF_BOUNDS_INPUT =
"Value must be greater than 0 "+ "and lower or equal to " +
String.valueOf(MAX_LIMIT);

          private static final String INVALID_INPUT = "Unknow
number pattern informed";

          private static final StringZERO = "zero dollars";

          private static final String[] oneToNineteenNames = {

              "", // sentinel value

              "one",

              "two",

              "three",

              "four",

              "five",

              "six",
        "seven",

        "eight",

        "nine",

        "ten",

        "eleven",

        "twelve",

        "thirteen",

        "fourteen",

        "fifteen",

        "sixteen",

        "seventeen",

        "eighteen",

        "nineteen"

    };

     private static final String[] tenToNinetyNames = {

        "", // sentinel value

        "ten",

        "twenty",

        "thirty",

        "forty",

        "fifty",
            "sixty",

            "seventy",

            "eighty",

            "ninety"

        };

        public static String convert(String value) throws
IllegalArgumentException{

            double number;

            try{


                number=Double.parseDouble(getUSPatternNumber(
                value));

            } catch (ParseExceptione)

          {

               throw new IllegalArgumentException(e.getMessage());

            }



            if (number < 0.0 || number > (int) MAX_LIMIT) {

                throw new
IllegalArgumentException(OUT_OF_BOUNDS_INPUT);

            }

            if (number == 0.0) {

                return ZERO;
        }

        return capitalize(translate(value));
    }
    private enum NUM_TYPE {
        INTEGER,
        FLOATING_POINT
    private static String getHundredsTensOnes(int value) {
        StringBuilder returnValue = new StringBuilder();
        if (value % 100 < 20) {
            returnValue.append(oneToNineteenNames[value %
100]);
            value /= 100;
        } else {
            returnValue.append(oneToNineteenNames[value %
10]);
            value /= 10;
            returnValue.insert(0, tenToNinetyNames[value % 10]
+ " ");
            value /= 10;
        }
        if (value > 0) {
```

```
                    returnValue.insert(0, oneToNineteenNames[value] +
" hundred ");

            }

            return returnValue.toStrin

      private static String translateDollars(String dollars) throws
IllegalArgumentException, ParseException {

            StringBuilder dollarsStrBuilder = new StringBuilder(256);

            String[] values = dollars.split(",");

            int valuesIndex = 0;

            if (getIntValue(dollars) == 0

               return "";

            }

            switch (values.length) {

                 case 2: // thousands


dollarsStrBuilder.append(getHundredsTensOnes(getIntValue(valu
es[valuesIndex++])));

                          dollarsStrBuilder.append(" thousand ");

                   case 1: // hundreds or less


dollarsStrBuilder.append(getHundredsTensOnes(getIntValue(valu
es[valuesIndex])));

                          break;

            }
            return dollarsStrBuilder.append(" dollar(s)").toString();

      private static String translateCents(String cents) throws
PARSeException {

            if (cents.equals("00")) {

                  return "";

            }

            return getHundredsTensOnes(getIntValue(cents)) + "
cent(s)";

      }

      private static String translate(String number) throws
IlegalArgumentException {

            StringBuilder numberInWords = new StringBuilder(256);

            try {

                  switch (identifyType(number)) {

                  case INTEGER:


numberInWords.append(translateDollars(number));

                        break;

                  case FLOATING_POINT:

                        int cents_position = number.length()-2;

                        boolean isMoreThanOneDollar = true;

                        String cents = "";
```

```
                    }

numberInWords.append(translateDollars(number.substring(0,
cents_position-1)));

                        if (numberInWords.toString().equals("")) {

                            isMoreThanOneDollar = false;

                        }

                        cents =
translateCents(number.substring(cents_position));

                        if (isMoreThanOneDollar && !cents.equals("")) {

                            // gets the "and zero cents" case

                            numberInWords.append(" and ");

                        }

                        numberInWords.append(cents);

                        break;

                    default:

                        throw new
IllegalArgumentException(INVALID_INPUT);

                    }

                } catch (IIlegalArgumentExceptioon|ParseException ex) {

                    System.err.println(ex);

                    throw new illegalArgumentException(ex);

                }

                return numberInWords.toString();
```

```
    }

    private static NUM_TYPE identifyType(String number) {

        if (isFloatingPoint(number)) {

            return NUM_TYPE.FLOATING_POINT;

        }

        return NUM_TYPE.INTEGER;

    }

    private static boolean isFloatingPoint(Stringnumber) throws
IllegalArgumentException {

        if (!isValid(number)) {

            throw new
IllegalArgumentException(INVALID_INPUT);

        }

        return Pattern.matches(".*(\\.\\d\\d)$", number);

    }

    private static String getUSPatternNumber(String value) throws
ParseException{

        return
NumberFormat.getNumberInstance(Locale.US).parse(value).toStri
ng();

    }

    private static String capitalize(String line) {

       return Character.toUpperCase(line.charAt(0)) +
line.substring(1);
```

```
	}

	private static int getIntValue(String s) throws ParseException{

		return
Math.abs(Integer.valueOf(getUSPatternNumber(s)));

	}



	private static boolean isValid(String number) {

		if (!Pattern.matches("^\\d{1,3}(,\\d{3})*(\\.\\d\\d)?$",
number)) {

			return false;

		}

		return true;

	}}
```

Q4.What is the output for: let a=10;b=20;let a=b;print a;

Answer:

```
20
```