Question 1a. Let us design a class bankAccount. A bank account has an account number. The bank gives each account a different, unique number. Each instance of this class maintains one account with an owner, an account number and current balance.

Normally, the account numbers start with some +ve integer and keep on increasing as the new accounts are created. We need a way to assign a new account number to each instance as it is created. A new account can be created by giving the owner's name and an initial amount.

Nobody should be able to manipulate instance variables directly. Methods must be provided to access (i) name of the owner (ii) account number (iii) current balance, and (iv) deposit money in the account. [15]

```
class bankAccount{
  private static int nextAccountNumber = 1; //next account number; It must be a static variable  2 marks
  private String person;  //the account owner; must be private                                   1 mark
  private int number; //The account number; must be private                                     1 mark
  private double balance; //the balance in the account; must be private                         1 mark

  //constructor method
  bankAccount(String p, double b){
        person = p;                                                  1 mark
        balance = b;                                                 1 mark
        number = nextAccountNumber;                                  1 mark
        nextAccountNumber += 1;                                      2 marks
  }

  public int getNumber(){return number;}   //get account number                                 1 mark
  public String getName() {return person;}    //get name of the account holder                 1 mark
  public double getBalance() {return balance;}   //get current balance                         1 mark
  public void deposit(double a){balance += a;}   //deposit money in the account                2 marks
}
```

Question 1b. What does following method do? Explain. Use backside of the page. [5]

```
public static int myFunction1(int x){
    int integer1 = 0;
    int integer2;
    if(x >= 0)
         integer2 = 1;
     else{
         integer2 = -1;
         x *= integer2;
     }
     while(x > 0){
         integer1 = integer1 * 10+ (x%10);
         x = x/10;
     }
     return (integer2 * integer1);
  }
```

The method takes an integer and reverses it. However, it preserves the sign. For example -123 will become -321.

2 marks if answer is correct
5 marks if answer is supported with an example

Question 2. Assume that we have a square whose corners have coordinates (0,0), (2,0), (2, 2), and (0, 2). We have a random number generator which can randomly generate a real number between 0 and 1 (both inclusive). Describe your algorithm and give a method which will estimate value of π using this information. [20]

```
/* It is a simple question in computing probability.          12 marks for the description
 * Generate a large number of points randomly which fall
 * in the square and compute how many are at a distance
 * <= 1.0 from the centre. The coordinates of the centre are (1,1)
 * The ratio of points at distance<=1.0 and the total generated
 * points gives a value which should be pi/4.
 * This is the ratio of the area of circle of radius unit 1
 * enclosed in a square of side unit 2.
 * */
import java.util.*;

class q2{
  public static void main(String args[]){
     double pi;
     Random generator = new Random();
     double x, y, dist;                                  2 marks for the declaration
     int i,inCircle;
     inCircle=0;


     for (i=1; i<=1000000; i++){      2 marks for the loop if the number of iterations are large
        x = 2*generator.nextDouble();                  2 marks for the statement
        y = 2*generator.nextDouble();
        dist = Math.sqrt((x-1)*(x-1)+(y-1)*(y-1));
        if (dist <= 1.0) inCircle += 1;
     }

     pi = 4.0 * inCircle / 1000000.0;                  2 marks; it should not be an integer division
     System.out.println(pi);
  }
}
```

Question 3a. Suppose a1 and a2 are two arrays of type int[]. How will you check whether the contents of two arrays are equal? Describe your strategy and complete the method below:
public static boolean equals(int a1[], int a2[]){ … }

Make sure that your solution is efficient. [10]

```
public static boolean equals(int a1[], int a2[]){
        if (a1==a2) return true;                          |
        if ((a1==null) || (a2==null)) return false;       |4 marks
        if (a1.length!=a2.length) return false;           2 marks
        for (int i=0;i<a1.length; i++)                    3 marks for the loop
                if (a1[i]!=a2[i]) return false;
        return true;                                      1 mark
}
```

The two arrays are considered equal if both arrays contain the same number of elements, and all corresponding pairs of elements in the two arrays are equal. Two objects e1 and e2 are considered equal if (e1==null ? e2==null : e1.equals(e2)). In other words, the two arrays are equal if they contain the same elements in the same order. Also, two array references are considered equal if both are null. **(Reference: Sun Java Docs)**

Question 3b. A list of item names together with prices can be maintained in two arrays as follows:

String items = new String[1000];
double prices = new double[1000];

Array items and prices are *parallel* arrays. If we have more data like weight of an item, price for buying two items at a time, we would add more parallel arrays.

Is this a good strategy to maintain information? Justify your answer. If you do not like this style then suggest an alternative and again justify your choice. [10]

```
/* Parallel arrays are fast to implement. However,
 * this in not a good programming style.
 * 1. Methods which operate on the parallel arrays
 * must have all the arrays as parameters.
 * 2. A method which deals with just one item must
 * be passed all the corresponding elements.
 * 3. Once a program has been written, it will be very
 * difficult to change if a new parallel array containing
 * a new property has to be added.
 * 4. There is always a danger of accessing items using
 * an index variable and its properties using another
 * index variable.
 *
 * define a class whose instances are items and their
 * associated properties.
 */
class item {
        //name and properties of items
        private String item;
        private double price;

        //variables for other properties
        //constructor methods
        //methods to access variables
}
class q3b{
  public static void main(String args[]){
          //using parallel arrays
          String items[] = new String[1000];
          double prices[] = new double[1000];
          //using class
          item x = new item();
          item y = new item();
          //code to get information about
          //properties of x, y, etc.
  }
}
```

6 marks for giving reasons
At least two reasons must be given

3 marks for giving solution

1 mark for the code

Question 4a. The following is a program intended for performing a binary search on a sorted array. Method bsrch returns an index for the key if the key is found and otherwise returns -1.

Is this method correct (there are no compilation errors!)? If not then show the errors with the help of an example and correct them. [8]

```
class q4a{
  public static int bsrch(int a[], int key) {
    int left = 0;
    int right = a.length -1;
    int mid;
    while (left < right) {
       mid = (left + right)/2;
       if (a[mid] == key) return mid;
       if (a[mid] < key) left = mid+1;
         else right = mid -1;
    }
    return -1;
}
```

The relational operator in the while loop should be replaced by <=

Case when relational operator is < **4 marks for identifying the error with an example**
Assume that a is {1,3} and key is 3
left=0, right=1, mid=0 a[mid] =1 which is < 3 therefore left=1
Now condition left<right is false and the method returns -1 which is wrong

Case when relational operator is <= **4 marks for correcting error and showing that it works**
Again assume that a is {1,3} and key is 3
left=0, right=1, mid=0a[mid]=1 which is < 3 therefore left=1
condition left<=right is true; therefore, mid=1
a[1]=3 which is same as key, therefore return 1 which is the correct answer

Question 4b. Modify the program so that it returns the left most occurrence of the key in the array in case there are duplicate elements. Describe your algorithm.[12]

```
public static int lmbsrch(int a[],int key) {
   int left =0;
   int right = a.length -1;
   int mid;
   while (left < right) {
      mid = (left + right)/2;
      if (a[mid] == key) right = mid;   //continue searching in the left if key matches an element
                                                                                    6 marks
       else if (a[mid] > key) right = mid-1;    //search in the left as in binary search    2 marks
         else left = mid +1;              //search in the right as in binary search          1 mark
   }

   if (a[right] == key) return right;     //return index if cannot find any more left occurrences  3 marks
         else return -1;                  //return with failure otherwise
}
```

Question 5. We had discussed the tower of Hanoi problem. A programmer has modified our algorithm and has produced the following recursive method:

```
public static void shift(int n, char source, char target, char using){
        int mid;
        if (n==0) return;
        if (n==1) System.out.println(source+" -> "+target);
          else if (n > 1) {
            mid=n/2;
            shift(n-mid,source,using,target);
            shift(mid,source,target,using);
            shift(n-mid,using,target,source);
   }
}
```

Does it produce the desired moves? Justify your answer. How many and what moves does it generate for 3 and 4 disks respectively? [20]

For 3 disks it produces following 7 moves: **4 marks**
A->C; A->B; C->B; A->C; B->A; B->C; A->C
which are correct.

For 4 disks it produces following 9 moves: **6 marks**
A->C; A->B; C->B; A->B; A->C; B->C; B->A; B->C; A->C
which are not the correct moves.

This does not produce correct results. Assume a case of 4 disks d1, d2, d3, d4 (with d1 being the largest and d4 being the smallest), and the three sticks A, B and C.
First recursive call to method shift will be shift(2,A,B,C) which will shift d3, d4 to B
Then it will call shift(2,A,C,B) which will attempt to shift d1,d2 from A to C.
However, in this process it will have to make moves: move d2 from A to B, move d1 from A to C, move d2 from B to C.
This leads to wrong configuration because after the first move (described in the above line) stick B will have d3 at the bottom, d4 in the middle and d2 on the top. As d2 is larger than d4 the configuration is incorrect. **10 marks for explanation**

Question 6. We have used string rotation for encryption. Let us again attempt to left rotate characters of a string s by a number n. Assume that value of n <= s.length().
We are given a method public static String reverse(String s, int start, int end)
which reverses substring of s between indices start and end (both inclusive). For example reverse("abcdef",1,3) returns "adcbef". Write a method which rotates a string using the reverse method. Give brief justification for your solution. Complete the method rotate.

You may find following identities useful: $(A^R)^R = A$ and $(A^RB^R)^R = BA$
Where A and B are strings and A^R is reverse string of A. [20]

```
public static String rotate(String s, int n){
        String r="";
        r = reverse(s,0,n-1);
        r = reverse(r,n,r.length()-1);
        r = reverse(r,0,r.length()-1);
        return r;
}
```

The key to the solution is to realize that left rotation of a string of length m by n places is equivalent to splitting the string in substrings of length n and length m-n and changing their order.

For example, if we have a string s="abcdefghij" of length 10 and wish to left rotate it by 3 places then it can be split into two substrings of length 3 which gives "abc" and length 7 which gives "defghij" and changing their order in the original string which gives "defghijabc"

Using the identity given above left rotation by n is equivalent to:
- reverse substring of length n
- reverse substring of length m-n
- reverse the whole string to get left rotated string

15 marks for the solution
5 marks for the code; check indices carefully

Question 7a. Write a method FindMaxOccurElement which takes a sorted array, and returns the element with the highest occurrence. In case of tie, the method returns the largest element. (DO NOT use another array to store the frequencies of each number.) [10]

```
class FindMaxFreq{
  public static int FindMaxOccur(int [] A){
    int max_count = 1, max_elem = A[0];
    int i, count=1;

    for (i=1; i < A.length; i++){
      if (A[i]==A[i-1]) count++;
        else if (count >= max_count)
           { max_count = count; max_elem = A[i-1]; count = 1;}
         else {count = 1;}
    }
    if (i==A.length && count >= max_count) {
              max_count = count; max_elem = A[i-1];
    }
    System.out.println("Freq : "+max_count);
    return max_elem;
  }

  public static void main(String args[]){
  int data[] = {1,5,5,5,5,5,6,7,9,11,13,18,18,18,18,20,20,20,20,20,31,31,31,31};

   for (int i = 0; i <data.length; i++) System.out.print(data[i]+" ");
     System.out.println();
     System.out.println("Max Element : " +FindMaxOccur(data));
 }
}
```

Solution involves the following steps:

- init vars for max_occur element
- increment in case of equal element
- update max_occur vars if found
- re-init for next occur
- update max_occur in case last element is the max_occur

Grading scheme
Use of another array: No credit
Using more than n comparisons
(that means you are not using the fact that arrays are sorted): 2 marks
Correct solution with n comparisons: 10 marks
Not correct solution but in the right direction (you will get partial credit): maximum 6 marks

Question 7b. All arithmetic expressions require that parenthesis be properly balanced. The number of the left and the right parenthesis should be equal, and there must a matching right parenthesis for every left parenthesis.
Give a method which will check whether parentheses in an expression are well formed. What data structure will you use for solving this problem? You do not have to write Java program. It is sufficient to describe the method. [10]

A stack can be used to find out if the parenthesis in an expression is well formed or not

While input has some symbols
    If input is (then push it on the stack
    If input is) then pop a symbol from the stack (check whether stack is empty before popping)
        If stack is empty and there is nothing to be popped
            then there are more) and expression is not well balanced
    Continue to look at the input symbols by incrementing input pointer
End of while loop //at this point input has been exhausted
If stack is empty
    Then parenthesis are well balanced
    Else there are more (and expression is not well balanced

Grading scheme: Using counters without a check on negative values: no credit
Using counter with care for open parenthesis first: 10 marks
usingstack and the above algorithm: 10 marks
Using stack but the solution is incorrect (you will get partial credit): maximum 6 marks

```
class q7b{
   public static void main(String args[]){
   Stack x = new Stack(100);
   String s = "(()()()())";
   char c,d;
   int i=0;
   while (i<s.length()){
     c = s.charAt(i);
     if (c == '(') x.push(c);
       else if (c==')'){
            if (x.isEmpty()){
                 System.out.println("More right parenthesis");
              return;}
             else
                 d=x.pop();
             }
         }
         i++;
   }
   if (x.isEmpty())
          System.out.println("balanced parenthesis");
   else System.out.println("More left parenthesis");
  }
}
```

Question 8a. Assume a string which contains only digits and blanks. It can have both leading and trailing blanks. An example is " 134 12 1 0 21 ".
We wish to extract all the integer numbers out of this string and print them. For example, in this case we will extract 134, 12, 1, 0, and 21.

Describe in English and/or Java how you will achieve this objective. [10]

```
/*
 while string has non blank characters
   - remove the blanks from the begining of the string
   - add a blank character at the end of the string
                to ensure that the string ends in a blank
   - find index i of the first blank in the string
   - convert s[0..i-1] into a string and store in n
   - print n
   - remove s[0..i-1] from s
*/

class q1b{
  public static void main(String args[]){
      String s = "   134  12    1   0   21  ";
         while (s.length() > 1){
            s = s.trim();
            s = s + " ";
            int i = s.indexOf(" ");
            int n = Integer.parseInt(s.substring(0,i));
            System.out.println(n);
            s = s.substring(i);
         }
  }
}
```

10 marks if the description/code is correct.
Ignore minor syntactic errors in the code

Question 8b. Remember the Pascal's Triangle?! It's rows contain binomial coefficients. If you recall we first generated this triangle by computing ${}^{n}C_{r}$. However, this method had very high time complexity. In the second method we stored information in a 2-D array and did just additions to compute the coefficients. This method was faster but required more storage. The code for the method is given below:

Can you modify this code to reduce storage requirement? Describe your strategy. [10]

```
class q2b{
  public static int[][] PascalTriangle1(int n){
     int a[][] = new int[n][n];
     for (int i=0; i<n; i++){
       a[i][0]=1;
       for (int j=1; j<=i; j++)
         a[i][j] = a[i-1][j-1] + a[i-1][j];
     }
     return a;
  }

  //Space saving PascalTriangle.
  //Saves nearly 50% space. Uses n(n+1)/2 instead of n*n

public static int[][] PascalTriangle2(int n){
     int a[][] = new int[n][]; //do not declare column size          4 marks
     for (int i=0; i<n; i++){
       a[i] = new int[i+1];  //declare just large enough array      2 marks
       a[i][0]=1;
       for (int j=1; j<i; j++) //condition has to be < and not <=   2 marks
                               //otherwise an exception will occur
         a[i][j] = a[i-1][j-1] + a[i-1][j];
       a[i][i]=1;  //last element has to be manually assigned 1; it can  not be computed
                                                                     2 marks
     }
     return a;
  }
```

Question 9a. Assume an array x which has been declared as follows: double x[] = new double[1000];

We had written a program to find minimum and maximum elements of x. The relevant code fragment is given below. Assuming x is filled with random numbers, how many comparisons does it take to find min and max? Can you reduce the number of comparisons? If yes then describe your strategy and modify the code. How many comparisons does the modified code take? [10]

```
//find min and max
min=x[0]; max=x[0];
for (int i=1; i<x.length; i++){
        if (x[i]<min) min=x[i];
        if (x[i]>max) max=x[i];
}

//use fewer comparisons
min=x[0]; max=x[0];
for (int i=1; i<x.length; i++){
        if (x[i]>max) max=x[i];
        else if (x[i]<min) min=x[i];
}
```

Original code takes ~2n comparisons **2 marks**

Reduce comparisons

- If a number is max then it cannot be min. **5 marks**
- If elements of x[] are randomly distributed then x[i]>max will fail 50% of the time. This will require a second comparison in the else clause. Therefore, total number of comparisons will be ~1.5N instead of ~2N (N is size of the array) **3 marks**

Question 9b. Design a Newton-Raphson iteration for solving $\sqrt[k]{c}$ (c>0 and k is +ve integer). [10]

$x = \sqrt[k]{c}$ ➔ $x^k = c$

$f(x) = x^k - c = 0$ **4 marks**

$f'(x) = kx^{k-1}$

Using Newton-Raphson method iterative equation $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ **3 marks**

We get $x_{n+1} = x_n - \frac{x^k - c}{k * x^{k-1}}$ **3 marks**

Which needs to be solved iteratively to get a root.