Code Review

maze.java

Your code was:

```java
import java.util.ArrayList;
import java.util.Arrays;

public class maze {
    public static void main(String[] args) {
//          System.out.println(maze_LR_list("",3,3));
        boolean[][] restriction={
                {true,false,true},
                {true,true,true},
                {true,true,true}
        };
        int[][] path = new int[restriction.length][restriction[0].length];
        maze_All_Print("",0,0,restriction,1,path);
    }

    static void maze_LR(String p, int r,int c){                               // ONLY LEFT RIGHT MOVEMT
        if(r==1 && c==1){
            System.out.println(p);
            return;
        }
        if (r>1){
            maze_LR(p+'D',r-1,c);
        }
        if (c>1){
            maze_LR(p+'R',r,c-1);
        }

    }
    static ArrayList<String> maze_LR_list(String p, int r, int c){                  // RESULT IN ARRAY
        ArrayList<String> arr=new ArrayList<>();
        if(r==1 && c==1){
            arr.add(p);
            return arr;
        }
        ArrayList<String> ans=new ArrayList<>();
        if (r>1){
            ans.addAll(maze_LR_list(p+'D',r-1,c));
        }
        if (c>1){
            ans.addAll(maze_LR_list(p+'R',r,c-1));
        }
        return ans;
    }
    static void maze_LRD(String p, int r,int c){                                     //BOTTOM RIGHT AND DIAGONA
        if(r==1 && c==1){
            System.out.println(p);
            return;
        }
        if (r>1){
            maze_LRD(p+'B',r-1,c);
        }
        if (c>1){
            maze_LRD(p+'R',r,c-1);
        }
        if (r>1&&c>1){
```

```
        maze_LRD(p+'D',r-1,c-1);
    }

}
static void maze_LRD_restriction(String p, int r,int c,boolean rest[][]){
    if(r==0 && c==0){
        System.out.println(p);
        return;
    }
    if(!rest[r][c]){
        return;
    }
    if (r>0){
        maze_LRD_restriction(p+'B',r-1,c,rest);
    }
    if (c>0){
        maze_LRD_restriction(p+'R',r,c-1,rest);
    }
    if (r>0&&c>0){
        maze_LRD_restriction(p+'D',r-1,c-1,rest);
    }

}
static void maze_AllD(String p, int r,int c, boolean [][] rest ){             //LEFT RIGHT DOWN UP AND BA
    if(r== rest.length-1 && c== rest[0].length-1){
        System.out.println(p);
        return;
    }
    if(!rest[r][c]){
        return;
    }

    rest[r][c]=false;

    if (r< rest.length-1){
        maze_AllD(p+'B',r+1,c,rest);
    }
    if (c< rest[0].length-1){
        maze_AllD(p+'R',r,c+1,rest);
    }
    if (c>0){
        maze_AllD(p+'L',r,c-1,rest);
    }
    if (r>0){
        maze_AllD(p+'U',r-1,c,rest);
    }
    if (r< rest.length-1 && c < rest[0].length-1){
        maze_AllD(p+'D',r+1,c+1,rest);
    }

    rest[r][c]=true;

}

static void maze_All_Print(String p, int r, int c, boolean[][] rest, int step,int[][] path){
    if(r== rest.length-1 && c== rest[0].length-1){
        path[r][c]=step;
        for(int[] arr:path){
            System.out.println(Arrays.toString(arr));
        }
```

```
            System.out.println(p);
            System.out.println();
            return;
        }
        if(!rest[r][c]){
            return;
        }

        rest[r][c]=false;
        path[r][c]=step;

        if (r< rest.length-1){
            maze_All_Print(p+'B',r+1,c,rest,step+1,path);
        }
        if (c< rest[0].length-1){
            maze_All_Print(p+'R',r,c+1,rest,step+1,path);
        }
        if (c>0){
            maze_All_Print(p+'L',r,c-1,rest,step+1,path);
        }
        if (r>0){
            maze_All_Print(p+'U',r-1,c,rest,step+1,path);
        }
        if (r< rest.length-1 && c < rest[0].length-1){
            maze_All_Print(p+'D',r+1,c+1,rest,step+1,path);
        }

        rest[r][c]=true;
        path[r][c]=0;
    }


}
```

Here is the review of your code: **Marks scored: 6/10**

Pros:

- Good use of functions to solve different parts of the problem.
- Code is modular and each function serves a unique purpose.

Cons (with marks deducted and reason):

- Deducted 2 points because the class name 'maze' does not follow Java naming conventions. Class names should be in title case and should describe what object the class represents.
- Deducted 1 point for the lack of comments. Comments should be used to describe what each function does and what the variables represent.
- Deducted 1 point for the lack of proper indentation and line breaks, which would make the code more readable.

subset.java

Your code was:

```
public class subset {
    public static void main(String[] args) {
//          prmtations("","abc");

    }
    static void skip(String p, String up){
        if(up.isEmpty()){
            System.out.println(p);
            return ;
        }
        char ch = up.charAt(0);

        if (ch=='a'){
            skip(p, up.substring(1));
        }else{
            skip(p+ch,up.substring(1));
        }
    }
    static String skips(String up){
        if(up.isEmpty()){
            return "";
        }

        char ch = up.charAt(0);

        if (ch=='a'){
            return skips(up.substring(1));
        }else{
            return ch + skips(up.substring(1));
        }
    }

    static String skipapple(String up){
        if(up.isEmpty()){
            return "";
        }



        if (up.startsWith("apple")){
            return skipapple(up.substring(5));
        }else{
            return up.charAt(0)+skipapple(up.substring(1));
        }
    }

    static void sub(String p, String up){
        if (up.isEmpty()){
            System.out.println(p);
            return;
        }

        char ch= up.charAt(0);

        sub(p,up.substring(1));
        sub(p+ch, up.substring(1));
    }
    static void subascii(String p, String up){
        if (up.isEmpty()){
            System.out.println(p);
            return;
        }

        char ch= up.charAt(0);

        subascii(p,up.substring(1));
        subascii(p+ch, up.substring(1));
        subascii(p+ (ch+0), up.substring(1));
    }



}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: 1. The code contains multiple methods showing a good understanding of method creation in Java. 2. The use of string manipulation and recursive function calling is commendable. 3. Logic for skipping certain patterns in the string (like "a" or "apple") is defined well.

Cons (with marks deducted and reason): 1. **(-2 marks)** The class name 'subset' does not follow Java naming conventions. It should start with an uppercase, e.g., 'Subset'. 2. **(-1 mark)** The code lacks comments which could make it hard for someone else to understand the purpose of each function. 3. **(-1 mark)** There are unused functions, and main function is not executing any operations, showing a lack of completeness in the code.

sudoku.java

Your code was:

```
public class sudoku {
    public static void main(String[] args) {
        int[][] board=new int[][]{{5, 3, 0, 0, 7, 0, 0, 0, 0},
        {6, 0, 0, 1, 9, 5, 0, 0, 0},
        {0, 9, 8, 0, 0, 0, 0, 6, 0},
        {8, 0, 0, 0, 6, 0, 0, 0, 3},
        {4, 0, 0, 8, 0, 3, 0, 0, 1},
        {7, 0, 0, 0, 2, 0, 0, 0, 6},
        {0, 6, 0, 0, 0, 0, 2, 8, 0},
        {0, 0, 0, 4, 1, 9, 0, 0, 5},
        {0, 0, 0, 0, 8, 0, 0, 7, 9}};

        if (solver(board)){
            show(board);
        }else{
            System.out.println("no");
        }

    }
    static boolean solve(int[][] board){
        int n= board.length;
        int col=-1;
        int row=-1;

        boolean empty=true;

        for (int r = 0; r < n; r++) {
            for (int c = 0; c < n; c++) {
                if (board[r][c]==0){
                    col=c;
                    row=r;
                    empty=false;
                    break;

                }
            }
            //if empty element is found in the row
            if (empty==false){
                break;
            }
        }
        if (empty==true) {
            return true;
        }

        //backtracking
        for (int numb = 1; numb <=9 ; numb++) {
            if(isSafe(board,row,col,numb)){
                board[row][col]=numb;
                if(solve(board)){
                    return true;
                }else{
                    //backtrack
                    board[row][col]=0;
                }
            }
        }
        return false;

    }

    private static void display(int[][] board) {
        for (int[] row : board){
            for(int num: row){
                System.out.print(num+" ");
            }
            System.out.println();
        }
    }

    static boolean isSafe(int[][] board, int row , int col , int num){
        //check full row
        for (int i = 0; i < board.length; i++) {
            if (board[row][i]==num){
                return false;
            }
        }

        //check full col
        for(int[] nums:board){
            if(nums[col]==num){
                return false;
            }
        }

        //check full box
        int sqrt=(int)(Math.sqrt(board.length));
        int rowStart= row-row%sqrt;
        int colStart = col-col%sqrt;

        for (int i = rowStart; i < rowStart+sqrt; i++) {
            for (int j = colStart; j < colStart+sqrt; j++) {
                if (board[i][j]==num){
                    return false;
                }

            }
        }
        return true;
```

```
}


static boolean solver(int[][] board){
    int n=board.length;
    int col= -1;
    int row = -1;
    boolean isempty=true;

    for (int r= 0; r <n ; r++) {
        for (int c = 0; c < n; c++) {
            if (board[r][c]==0){
                isempty=false;
                row=r;
                col=c;
                break;
            }
        }
        if(isempty==false){
            break;
        }

    }
    if (isempty == true) {
        return true;
    }

    for (int i = 1; i <= board.length; i++) {
        if (safe(board,row,col,i)){
            board[row][col]=i;
            if (solver(board)) {
                return true;
            }else{
                board[row][col]=0;
            }
        }

    }
    return false;
}

static void show(int[][]board){
    for (int[] row:board){
        for (int ele:row){
            System.out.print(ele+" ");
        }
        System.out.println();
    }

}
static boolean safe(int[][] board, int r, int c, int num){
    //check row
    for (int i = 0; i < board.length; i++) {
        if(board[r][i]==num){
            return false;
        }

    }
```

```
        //check col
        for (int i = 0; i < board.length; i++) {
            if(board[i][c]==num){
                return false;
            }
        }

        //check box
        int sqrt=(int)(Math.sqrt(board.length));
        int startrow=r-r%sqrt;
        int startcol=c-c%sqrt;
        for (int i = startrow; i < startrow+sqrt; i++) {
            for (int j = startcol; j < startcol+sqrt; j++) {
                if (board[i][j]==num){
                    return false;
                }
            }

        }
        return true;

    }


}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - Use of functions for modularity. - Algorithm implementation seems correct with the usage of backtracking.

Cons (with marks deducted and reason): - The filename 'sudoku.java' should start with an uppercase character as per Java naming standards. Deducted 1 mark. - Inconsistencies in uses of spacing and indentation can make your code hard to read. Deducted 1 mark. - The class name must start with a capital letter. Deducted 1 mark. - The code is repetitive and not efficiently optimized. For example, there are two display and safe functions. Deducted 1 mark. - Comments could be included to explain the logic in a better manner. Deducted 0.5 marks. - There is no usage of access modifiers making the code less secure. Deducted 0.5 marks.

NQueens.java

Your code was:

```
public class NQueens {
    public static void main(String[] args) {
        int n=5;
        boolean[][] board=new boolean[n][n];
        System.out.println(Queen_meri(board,0));

    }
//    static int queens(boolean[][] board, int row){
//        if (row==board.length){
//            display(board);
//            System.out.println();
//            return 1;
//        }
//
//        //placing queen and checking
//        int count=0;
//        for (int col = 0; col < board.length; col++) {
//            if (isSafe(board,row, col)){
```

```
//                  board[row][col]=true;
//                  count+=queens(board,row+1);
//                  board[row][col]=false;
//              }
//
//          }
//          return count;
//      }
//
//      private static void display(boolean[][] board) {
//          for(boolean[] bool : board){
//              for (boolean element:bool){
//                  if (element){
//                      System.out.print("Q ");
//                  }else{
//                      System.out.print("X ");
//                  }
//              }
//              System.out.println();
//          }
//      }
//
//      private static boolean isSafe(boolean[][] board, int row, int col) {
//          // check vertically
//          for (int i = 0; i < row; i++) {
//              if (board[i][col]){
//                  return false;
//              }
//          }
//
//          //check left diagonal
//          int maxLeft=Math.min(row,col);
//          for (int i = 0; i <= maxLeft; i++) {
//              if(board[row-i][col-i]){
//                  return false;
//              }
//          }
//          //check max right
//          int maxRight=Math.min(row, board.length-col-1);
//          for (int i = 0; i <= maxRight; i++) {
//              if (board[row-i][col+i]){
//                  return false;
//              }
//          }
//          return true;
//      }


    // KHUDSE

    static int Queen_meri(boolean[][] game,int r){
        if (r==game.length){
            show(game);
            System.out.println();
            return 1;
        }
        int sum=0;

        for (int c = 0; c < game.length; c++) {
            if (safe(game,r,c)){
                game[r][c]=true;
                sum+=Queen_meri(game,r+1);
                game[r][c]=false;
            }
        }
        return sum;
    }

    private static void show(boolean[][] game) {
        for (boolean[] bool:game){
            for (boolean ele:bool){
                if(ele){
                    System.out.print("Q ");
                }else{
                    System.out.print("X ");
                }
            }
            System.out.println();
        }
    }

    private static boolean safe(boolean[][] game, int r, int c) {
        //check upar
        for (int i = 0; i < r; i++) {
            if(game[i][c]){
                return false;
            }
        }
        //check left dia
        int left=Math.min(r,c);
        for (int i = 0; i <= left ; i++) {
            if(game[r-i][c-i]){
                return false;
            }
        }

        //check right dia
        int right=Math.min(r, game.length-1-c);
        for (int i = 0; i <=right ; i++) {
            if (game[r-i][c+i]){
                return false;
            }

        }
        return true;
    }




}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - The file name 'NQueens.java' is appropriate and adheres to Java naming convention. - The code logic for solving N-Queens problem is correct. - Uses boolean array for holding the chessboard which optimizes memory usage.

Cons (with marks deducted and reason): - Inefficient use of comments. Major part of the code is commented out without explanation (-1 mark). - Function and variable names like 'Queen_meri', 'game', 'r', 'c', 'bool', 'ele' etc. are unclear and do not adhere to naming conventions (-1 mark). - Has hard-coded the value of 'n' (the size of the chessboard), which should ideally be an input parameter (-1 mark). - Lack of code modularity. Functions could be split into smaller, reusable parts (-1 mark).

recursion.java

Your code was:

```
import java.util.Arrays;

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

//        QUESTION 1


//        int[] input= {1,2,3,4,5};
//        trisum(input);


//        QUESTION 2

//        int [] arr = {1, 4, 45, 6, 10, -8};
//        minmax(arr,0,arr[0],arr[0]);


//        QUESTION 3          (GOOD QUESTION)

//        int[] input={-1,0,3,5,9,12};
//        int target=12;
//        int start=0;
//        int end=input.length-1;
//        binary(input,target,start,end);

//       QUESTION 4 (kuch naya pata chala)
//        String input="geekS";
//        upper(input,0);


//        QUESTION 5 ( learnt effective eay of reversal)
//        String [] arr = {"H","a","n","n","a","h"};
//        reverse(arr,arr.length-1,0);

//        QUESTION 6

//        int limit=5;
//        print(limit,1);
    }

    static void trisum(int[] arr) {
        if (arr.length == 1) {
            System.out.println(Arrays.toString(arr));
            return;
        }
        int num = 0;
        int[] ans = new int[arr.length - 1];
        for (int i = 1; i < arr.length; i++) {
            num = arr[i] + arr[i - 1];
            ans[i - 1] = num;
        }
        System.out.println(Arrays.toString(ans));
        trisum(ans);
    }
    static void minmax(int[] arr, int i, int max, int min) {
//        min = arr[0];
//        max = arr[0];
        if (arr[i]>max){
            max=arr[i];
        }
        if (arr[i]<min){
            min=arr[i];
        }
        if (i==arr.length-1){
            System.out.println("MAX = "+ max);
            System.out.println("MIN = "+ min);
            return;
        }
        minmax(arr,i+1, max,min);

    }
    static void binary(int[] arr, int target, int start, int end){
        if(start>end){
            System.out.println(-1);
            return;
        }
        //mid
        int mid= start + (end - start) / 2;
        if (arr[mid]>target){
//                end=mid-1;
            binary(arr,target,start, mid-1);                                // NOTE

        }else if (arr[mid]<target){
//                start=mid+1;
            binary(arr,target,mid+1, end);

        }else if(arr[mid]==target){
            System.out.println(mid);
            return;
        }

    }
    static void upper(String var,int i){
        if (Character.isUpperCase(var.charAt(i))){                          // NEW INBUILT FUNCTION
            System.out.println(var.charAt(i));
            return;
        }
        upper(var,i+1);
    }

    static void reverse(String[] a,int start, int end){
//        String[] ans=new String[a.length];
//        if (start>=0){
//            ans[end]=a[start];
//            reverse(a,start-1,end);
//        }else{
//            System.out.println(Arrays.toString(ans));
//            return;
//        }
        if(start>=end){                                                      //NOTE
            String temp=a[start];
            a[start]=a[end];
            a[end]=temp;
            reverse(a,start-1,end+1);
        }else{
            System.out.println(Arrays.toString(a));
        }

    }
    static void print(int n,int i){
        if (i>n){
            return;
        }
        System.out.print(i + " ");
        print(n,i+1);
    }

}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - The code is structured and well formatted. - Clear implementation of various methods.

Cons (with marks deducted and reason): - FileName (deducting 1 mark) - The filename 'recursion.java' is not in accordance with Java class naming conventions. It should start with an uppercase letter. - Code Commenting (deducting 1 mark) - The code lacks sufficient comments describing what each piece of code or method is doing, making it difficult for others to understand the code. - Handling of Edge Cases (deducting 1 mark) - There is no error handling or checks incorporated into the code for undesired input/her scenarios. - Calling of functions (deducting 1 mark) - There is no function calling in the code. All function calls are commented which results to an untested program.

Nknights.java

Your code was:

```
public class Nknights {
    public static void main(String[] args) {
        int n=4;
        boolean[][] board=new boolean[n][n];
        System.out.println(knight(board,0));

    }
    static int knight(boolean[][] board,int row){
        if (row== board.length){
            display(board);
            System.out.println();
            return 1;
        }
        int count=0;

        for (int col = 0; col < board.length; col++) {
            if(isSafe(board,row,col)){
                board[row][col]=true;
                count+=knight(board,row+1);
                board[row][col]=false;
            }
        }
        return count;

    }

    private static boolean isSafe(boolean[][] board, int row, int col) {
        //check upar right
        if(isValid(board,row-2,col+1)){
            if (board[row-2][col+1]){
                return false;
            }
        }

        //check upar left
        if(isValid(board,row-2,col-1)){
            if(board[row-2][col-1]){
                return false;
            }
        }

        //check one upar right
        if (isValid(board,row-1,col+2)){
            if (board[row-1][col+2]){
                return false;
            }
        }

        //check one upar left
        if (isValid(board,row-1,col-2)){
            if (board[row-1][col-2]){
                return false;
            }
        }
        return true;
    }

    private static boolean isValid(boolean[][] board, int row, int col) {
        if(row>=0 && row<= board.length-1 && col>=0 && col<= board.length-1 ){
            return true;
        }
        return false;
    }

    private static void display(boolean[][] board) {
        for(boolean[] bool:board){
            for (boolean ele:bool){
                if(ele){
                    System.out.print("K ");
                }else{
                    System.out.print("x ");
                }
            }
            System.out.println();
        }
    }
}
```

Here is the review of your code: **Marks scored: 7/10**

Pros: - The code is simple and easy to understand - Good usage of main and helper functions - It makes effective use of recursion

Cons (with marks deducted and reason): - The class name 'Nknights' is not according to Java convention (-1). It should start with an uppercase letter and use CamelCase. - Hardcoding of value for 'n' is not recommended (-1). Use command line arguments or user input instead. - The absence of comments makes code interpretation hard (-1). A clear explanation of the methods, their parameters and return types would improve code readability.

patterns.java

Your code was:

```java
package assignments;

public class patterns {
    public static void main(String[] args) {
        int num=5;
        p14(num);

    }
    static void p2(int n){
        for (int row = 1; row <= n; row++) {
            for (int clm = 1; clm <= row; clm++) {
                System.out.print("*");
            }
            System.out.println("");
        }


    }

    static void p4(int n){
        for (int row = 1; row <= n; row++) {
            for (int clm = 1; clm <= row; clm++) {
                System.out.print(clm);
            }
            System.out.println("");
        }
    }

    static void p6(int n){
        for (int row = 1; row <= n; row++) {
            for (int clm = n-row; clm >=0; clm--) {
                System.out.print(" ");
            }
            for (int clm = 1; clm <= row; clm++) {
                System.out.print("*");
            }
            System.out.println("");
        }
    }
    static void p8(int n){
        for (int row = 1; row <= n; row++) {
            for (int clm = n - row; clm >= 0; clm--) {
                System.out.print(" ");
            }
            for (int clm = 1; clm <= (2*row)-1; clm++) {
                System.out.print("*");

            }
            System.out.println("");
        }
    }
    static void p10(int n){
        for (int row = 1; row <=n ; row++) {
            for (int clm = n - row; clm >= 0; clm--) {
                System.out.print(" ");
            }
            for (int clm = 1; clm <= row; clm++) {
                System.out.print("* ");

            }
```

```
                System.out.println("");
            }
        }

        static void p12(int n){
            for (int row = 1; row <=n ; row++) {
                for (int clm =1; clm <=row; clm++) {
                    System.out.print(" ");
                }
                for (int clm = n; clm >=row; clm--) {
                    System.out.print("* ");
                }
                System.out.println("");
            }
            for (int row = 1; row <=n ; row++) {
                for (int clm = n-row; clm >=0 ; clm--) {
                    System.out.print(" ");
                }
                for (int clm = 1; clm <=row ; clm++) {
                    System.out.print("* ");
                }
                System.out.println("");
            }
        }
        static void p14(int n){
            for (int row = 1; row <= n; row++) {
                if (row==1){
                    for (int clm = 1; clm <= (2*n)-1; clm++) {
                        System.out.print("*");
                    }
                }else{
                    for (int clm = 1; clm <=row-1 ; clm++) {
                        System.out.print(" ");
                    }
                    System.out.print("*");
                    for (int clm = ((2*n)-1)-(2*row); clm >=1; clm--) {
                        System.out.print(" ");
                    }
                    if(row!=n){
                        System.out.print("*");
                    }


                }
                System.out.println("");

            }
        }
}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - The code contains distinct modules used for generating different patterns, which is easy to understand. - Usage of looping constructs and printing methods is straightforward and correct.

Cons (with marks deducted and reason): - Deducted 2 points for the filename and class name. According to Java's naming conventions, both class names and filenames should start with a capital letter. Therefore, 'patterns' should be 'Patterns'. - Deducted 1 point for the naming of the methods. Method names should be verbs and express the operation that the method performs. Therefore, the p2, p4, p6... names are not ideally chosen according to the convention. - Deducted 1 point for not including comments in the code. Good code should always include comments for readability and improved understanding.

functions.java

Your code was:

```
package assignments;

import java.util.*;
public class functions {

    // to pass assignments.arrays to a function use [(data type)...v]
    // make sure the above thing is in the last of the arguments

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
//        int num1=sc.nextInt();

//        int num2=sc.nextInt();
//        System.out.println("sum = " + sum(num1,num2));
//        System.out.println("product = "+ prod(num1,num2));
//        prime(num1,num2);
//        grade(num1);
//        fact(num1);
//        palind(num1);
        pyth();
    }
    static int sum(int a, int b){
        return a+b;
    }
    static int prod(int a , int b){
        return a*b;
    }
    static void prime(int a, int b){

        int max=0;
        if (a>b){
            max=a;
        }else {
            max=b;
        }

        for (int i = 1; i <max ; i++) {
            if (a%i==0){
                System.out.println(a + "is composite");
            } else{
                System.out.println(a+" is prime");
                break;
            }
            if (b%2==0) {
                System.out.println(b +" is prime");

            }else{
                System.out.println(b+"is prime");
                break;
            }
        }

    }

    static void grade(int a){
        if (a<=100 && a>=91){
            System.out.println("AA");
        } else if (a>=81 && a<=90) {
            System.out.println("AB");
        }else if (a>=71 && a<=80) {
            System.out.println("BB");
        }else if (a>=61 && a<=70) {
            System.out.println("BC");
        }else if (a>=51 && a<=60) {
            System.out.println("CD");
        }else if (a>=41 && a<=50) {
            System.out.println("DD");
        }else if (a<=40) {
            System.out.println("FAIL");
        }
    }

    static void fact(int a){
        int fac=1;
        if(a==1 ||a==0){
            System.out.print("1");
        }else{

        for (int i = 0; i < a; i++) {
            int b = (a - i);
            if (i == (a - 1)) {
                System.out.print(b + " = " + fac);
            } else {
                System.out.print(b + " * ");
                fac = fac * b;
            }

            }


        }
    }

    static void palind(int a) {
        int n,rem,sum=0,c;
        c=a;
        while (a>0){

            rem=a%10;
            sum=(sum*10)+rem;
            a=a/10;


        }
        if(sum==c){
            System.out.println("palindrome");
        }else{
            System.out.println("not palindrome");
        }
    }
    static void pyth(){
        //to take assignments.arrays as input parameter....
        Scanner sc=new Scanner(System.in);
        System.out.print("enter total number of num u wanna enter : ");
        int lim=sc.nextInt();
        int [] arr=new int[lim];
        for (int i = 0; i < lim; i++) {
            System.out.print("Enter ur number : ");
            arr[i]=sc.nextInt();

        }
        int m=0,s=0;

        for (int i = 0; i < arr.length; i++) {
            int a =arr[i];
            for (int j = 0; j < arr.length; j++) {
                int b=arr[j];
                for (int k = 0; k < arr.length; k++) {
                    int c=arr[k];
                    if ((a*a)+(b*b)==(c*c)){
                        m+=1;
                    }else {
                        s+=1;
                    }

                }

            }
        }
        if (m>0){
            System.out.println("has pyth triplet");

        }else{
            System.out.println("no pytha triplet");
        }


    }

}
```

Here is the review of your code: **Marks scored: 6/10**

Pros:

- The code logic is right for most parts.
- The Scanner class is used effectively for user inputs.
- Comments included are useful for understanding the code.

Cons (with marks deducted and reason):

- Incorrect class name (should start with an upper case), deducting 1 mark for not following Java naming conventions.
- Several commented, unused lines of code, deducting 1 mark for maintaining clean code standards.
- Incorrect logic for calculating prime numbers, 1 mark deducted.
- The 'prime' function always says that the numbers are prime, even for composite numbers. For this misleading output, a mark is deducted.
- The file name 'functions.java' is too generic and does not describe the content sufficiently, half a mark deducted for this.

strings.java

Your code was:

```
package assignments;

import java.util.ArrayList;
import java.util.List;

public class strings {
    public static void main(String[] args) {
        String[] word1 = {};
        String[] word2 = {};
        System.out.println(concat(word1,word2));
    }
    static StringBuilder ipadress(String str){
        StringBuilder ans=new StringBuilder();
        for (int i = 0; i < str.length(); i++) {
            if(str.charAt(i)=='.'){
                ans.append("[.]");
            }else{
                ans.append(str.charAt(i));
            }

        }
        return ans;
    }
//    static StringBuilder insertion(String str, int[] arr){
////        StringBuilder ans =new StringBuilder();
////        for (int i = 0; i < arr.length; i++) {
////            ans.insert(i,'a');
////        }
////        for (int i = 0; i < arr.length; i++) {
////            ans.insert(arr[i],str.charAt(i));
////        }
////        return ans;
//        char[] chars=str.toCharArray();
//        List<Character> characterList=new ArrayList<>();
//        for (char c : chars){
//            characterList.add(c);
//        }
//    }

    static StringBuilder interpret(String comm){
        StringBuilder stb = new StringBuilder();
//        List<Character>chars=new ArrayList<>();
//        char[] chars1=comm.toCharArray();
//        for(char c : chars1){
//            chars.add(c);
//        }
//        for (int i = 0; i < chars1.length ; i++) {
//            if (chars1[i] == 'G') {
//                stb.append('G');
//            }else if(chars1[i]==')' && chars1[i-1]=='(' ){
//                stb.append('o');
//            }else if (chars1[i]==')'&& chars1[i-1]=='l' && chars1[i-2]=='a'&& chars1[i-3]==')'){
//                stb.append("al");
//            }
//        }
        for (int i = 0; i < comm.length(); i++) {
            if (comm.charAt(i)=='G'){
                stb.append("G");
            }else if (comm.charAt(i)=='('){
                if (comm.charAt(i+1)==')'&& i+1<comm.length()){
                    stb.append("o");
                }else{
                    stb.append("al");
                }
            }
        }
        return stb;
    }
    static StringBuilder sentrnceSort(String str){
        StringBuilder ans = new StringBuilder(str.length());
        int a=0;
        for (int i = 0; i < str.length(); i++) {
            if (str.charAt(i)==' '){
                for (int j = a; j <=i ; j++) {
                    ans.append(str.charAt(j));

                }
            }

        }
        return ans;
    }

    static Boolean concat(String[] word1,String[] word2 ){
        StringBuilder s1=new StringBuilder();
        StringBuilder s2=new StringBuilder();
        for (int i = 0; i < word1.length; i++) {
            for (int j = 0; j < word1[i].length(); j++) {
                s1.append(word1[i].charAt(j));
            }
        }
        for (int i = 0; i < word2.length; i++) {
            for (int j = 0; j < word2[i].length(); j++) {
                s2.append(word2[i].charAt(j));
            }
        }
        return s1.toString().equals(s2.toString());                              //STRING BUILDER KO STRING MEI

    }
//    String concatWord1 = String.join("", word1);                               // VERY GOOD SHORTCUT TO CONCAT
//    String concatWord2 = String.join("", word2);
//        return concatWord1.equals(concatWord2);
}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - Good use of StringBuilder for string operations which is more efficient than string concatenation. - Code is clean and well-spaced which enhances readability.

Cons (with marks deducted and reason): - -1 for the filename `strings.java`, which should start with a capital letter as per Java's naming convention. - -1 for unused or commented out code. This obstructs readability and makes the code messy. - -1 for not handling out of bound situations in the `interpret` method, if the string starts with '(', it checks `comm.charAt(i+1)` which can throw ArrayIndexOutOfBounds exception. - -1 for lack of comments to explain logic of the code. Good practice to add them for complex sections.

flowcharts.java

Your code was:

```
package assignments;

import java.util.*;

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

 // 1. WAP TO CHECK FOR LEAP YEAR
```

```
          Scanner sc=new Scanner (System.in);
        // System.out.print("ENTER THE YEAR : ");
        // int year=sc.nextInt();

        // if(year/4==0){
        //     System.out.println("LEAP YEAR");
        // }else{
        //     System.out.println("NOT leap year");
        // }

// 2. WAP to give sum of 2 nums

            // int frst=sc.nextInt();
            // int scnd=sc.nextInt();
            // System.out.println(frst+scnd);


// 3. WAP multiplication table of the given number

        // System.out.print("ENTER THE NUMBER U WANT MULTIPLICATION TABLE OF : ");
        // int num=sc.nextInt();
        // System.out.print("TILL WHERE DO U WANT THE TABLE : ");
        // int lim=sc.nextInt();

        // for (int i = 1; i <=lim; i++) {
        //     System.out.println(num + " * " + i + " = " + num*i);

        // }

// 4. WAP to find HCF and LCM................... (FOR REVIEW)

        // System.out.print("ENTER FIRST NUMBER : ");
        // int num1=sc.nextInt();
        // System.out.print("ENTER SECONF NUMBER : ");
        // int num2=sc.nextInt();
        // int hcf=0,max;

        // if(num1>num2){
        //     max=num1;
        // }else{
        //     max=num2;
        // }
        // //HCF

        // for (int i = 1; i <= max; i++) {
        //     if(i<=max && hcf%i==0){
        //         hcf/=i;
        //     }

        // }
        // System.out.print("HCF is " + hcf);

// 5. WAP summ till x

    // NEW FUNCTION Character.getNumericValue()

//         int sum=0;
//         boolean ver=true;
//
//         while(ver){
```

```
//            System.out.print("ENTER NUMBER : ");
//            char num=sc.next().trim().charAt(0);
//            if (num=='x'||num=='X'){
//                System.out.print("SUM = "+ sum);
//
//                ver=false;
//            }else{
//                int n=Character.getNumericValue(num);
//                sum+=n;
//            }










    }

}
```

Here is the review of your code: **Marks scored: 4/10**

Pros: 1. Code is commented to explain which section is solving what problem.

Cons (with marks deducted and reason):

1. **Improper Naming (-1)**: Filename `flowcharts.java` doesn't adhere to the standard Java convention. Class name should start with a capital letter and should be meaningful.
2. **Commented Code (-1)**: Large parts of the code are commented out. Code should be clean and ready to run without modification.
3. **Efficiency (-1)**: In the leap year calculation, the code checks if the year divided by 4 is 0, which will always be false. Common divisor in the HCF calculation is always 0. No check on input in sum calculation which can cause errors.
4. **Code Organization (-1)**: Too many problems solved within the `main` function. Implement divided pieces of functionality as separate methods in the class or in separate classes.
5. **Lack of Comments (-2)**: Proper comments explaining the functionality of each part of the code are lacking. It highlights lack of clear structure.

conditionals.java

Your code was:

```
package assignments;

import java.util.*;

public class conditionals {



    public static void main(String[] args) {


        // 1. WAP to find area of circle, rhombus, triangle, rectangle, quilateral trialngle, isosclestriangle
        Scanner sc = new Scanner(System.in);
//        System.out.print("AREAS YOU CAN FIND OUT :-\n" +
//                "1. CIRCLE\n" +
//                "2. RHOMBUS\n" +
//                "3. EQUILATEREAL TRIANGLE\n" +
//                "4. SCALENE TRIANGLE\n" +
//                "5. RECTANGLE\n" +
//                "6. PARALLELOGRAM\n");
//          System.out.print("ENTER NUMBER OF THE AREA YOU WANT TO FIND OUT : ");
//          int num=sc.nextInt();
//
//          if (num==1){
//                  circle();}
//          if (num==2) {
//              rhombus();
//          }
//          if (num==3) {
//              eqtri();
//          }
//          if (num==4) {
//              sctriangle();
//          }
//          if (num==5) {
//              rectangle();
//          }
//          if (num==6) {
//              parallelogram();
//          }
//
//
//
//      }
//
//      static void circle(){
//          Scanner sc=new Scanner(System.in);
//          System.out.println("enter radius of circle : ");
//          int r=sc.nextInt();
//          System.out.println("THE AREA OF THE CIRCLE IS : " + (2*3.14*r));
//      }
//
//      static void rectangle(){
//          Scanner sc=new Scanner(System.in);
//          System.out.print("length : ");
//          int l=sc.nextInt();
//          System.out.print("breadth : ");
//          int b=sc.nextInt();
//          System.out.println("THE AREA OF TEH RECTANGLE IS : " + (l*b));
//
//      }
//
//      static void eqtri(){
//          Scanner sc=new Scanner(System.in);
//          System.out.print("enter the length of sides : ");
//          int side= sc.nextInt();
//          System.out.println("The area of this equilateral triangle is : "+ ((1.73/4)*side*side));
//      }
//
//      static void rhombus(){
//          Scanner sc=new Scanner(System.in);
//          System.out.println("enter length of diagonal1");
//          int d2=sc.nextInt();
//          System.out.println("enter length of diagonal2");
//          int d1=sc.nextInt();
//          System.out.println("The area of the given rhombus is : " + (d1*d2));
//      }
//
//      static void parallelogram(){
//          Scanner sc=new Scanner(System.in);
//         System.out.print("enter length of base :");
//         int b=sc.nextInt();
//         System.out.print("enter length of height :");
//         int h=sc.nextInt();
//         System.out.print("The area of parallelogram is :" + (b*h));
//
//     }
//
//     static void sctriangle(){
//         Scanner sc=new Scanner(System.in);
//         System.out.print("enter length of base :");
//         int side1=sc.nextInt();
//         System.out.print("enter length of base :");
//         int side2=sc.nextInt();
//         System.out.print("enter length of base :");
//         int side3=sc.nextInt();
//
//         double s = ((double) (side1 + side2 + side3) / 2);
//         double area = Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
//
//         System.out.println("The area of the scalene triangle is: " + area);


        // WAP to find difference of sum and product of a numbers digits
//         System.out.println("ENTER A NUMBER : ");
//         int num=sc.nextInt();
//         int sum=0,prod=1;
//         while (num>0){
//             int ref=num%10;
//             sum=sum+ref;
//             prod=prod*ref;
//             num=num/10;
//
//
//
//
//     }
//         System.out.println("(product) "+prod+"-"+ " (sum) "+sum+" = "+ (prod-sum));


//WAP factors of a number

//         System.out.print("ENTER A NUMBER FOR FACTOR FINDING : ");
//         int num= sc.nextInt();
//         for (int i = 1; i < num; i++) {
//             if(num%i==0){
//                 if (i==1){
//                     System.out.print("The factors are : ");
//                 }else{
//                 System.out.print(i + ",");
//             }
//
//         }}


//WAP for int input and tell lasrgest number


//         int  max = 0;
//
```

```
//
//        while(true){
//            System.out.println("enter number : ");
//            int num = sc.nextInt();
//
//            if (num != 0) {
//                if (num > max) {
//                    max = num;
//                }
//            } else {
//                System.out.println(max);
//                break;
//
//            }
//    }
    }
}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: 1. The code is modular with each area calculation in a separate function. 2. Nice use of Scanner class to take inputs from user.

Cons (with marks deducted and reason): 1. Deducted 1 point because ‘conditionals’ is not a meaningful class name and goes against Java naming convention standards. ALWAYS use upper camel case (Pascal case) for class names. 2. Deducted 1 point because all the functions are static. This is not optimal and reduces the horizons of object oriented programming. 3. Deducted 1 point because most of the code is commented out and does not adhere to clean code standards. 4. Deducted 1 point as the code is not efficient. Instances of Scanner class are created multiple times, where only once is needed.

mergeSort.java

Your code was:

```
package assignments;

import java.util.ArrayList;
import java.util.Arrays;

public class mergeSort {
    public static void main(String[] args) {
        int[] arr = {1,2,3,0,0,0};
        int[] arr1= {2,5,6};
        int m=6,n=3;
        merging(arr,arr1,m,n);

//        arr=mergesort(arr);
//        System.out.println(Arrays.toString(arr));

    }
    static int[] mergesort(int[] arr){
        if (arr.length==1){
            return arr;
        }
        int mid=arr.length/2;
        int[] left = mergesort(Arrays.copyOfRange(arr,0,mid));
        int[] right= mergesort(Arrays.copyOfRange(arr,mid,arr.length));
        return merge(left,right);
    }
    static int[] merge(int[] first, int[] second){

        int i =0;
        int j=0;
        int[] mix=new int[first.length+second.length];
//        if (first.length<second.length){
//            while(i<first.length){
//                if(first[i]<second[i]){
//                    mix[i]=first[i];
//                    i++;
//                }else{
//                    mix[i]=second[i];
//                    i++;
//                }
//            }
//        }else{
//            while(i<second.length){
//                if(first[i]<second[i]){
//                    mix[i]=second[i];
//                    i++;
//                }else{
//                    mix[i]=first[i];
//                    i++;
//                }
//            }
//
//        }
        int k=0;
        while(i<first.length && j< second.length){
            if (first[i]<second[i]){
                mix[k]=first[i];
                i++;
            }else{
                mix[k]=second[j];
                j++;
            }
            k++;
        }
        while(i< first.length){
            mix[k]=first[i];
            i++;
            k++;
        }
        while(i< second.length){
            mix[k]=second[j];
            j++;
            k++;
        }
        return mix;
    }


    // QUESTIONS

    static void merging(int[] num1, int[] num2,int m,int n){
        int i=0;
        int j=0;


        while(i< m){
            if(num1[i]>num2[j] && j<=n){
                // swap
                int temp=num1[i];
                num1[i]=num2[j];
                num2[j]=temp;

                //add in last
                num1[num1.length- num2.length+j]=num2[j];
                i++;
                j++;
            }else{
                i++;
                j++;
            }
        }
        System.out.println(Arrays.toString(num1));
    }
}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - The code is logically correct and uncomplicated, using merge sort algorithm in an efficient manner. - Correct usage of arrays and ArrayList import.

Cons (with marks deducted and reason): - Minus 2: The naming of the file and class are against Java conventions. Classes should start with a capital letter. Changing `mergeSort` to `MergeSort` will be appropriate. - Minus 1: The code should contain comments to make it more readable and easy to understand. - Minus 1: Variables 'm' and 'n' are declared but not used which can lead to confusion. - In the merging() method, the second while loop has a condition where 'i' is compared to 'second.length', it should be 'j'.

firstjava.java

Your code was:

```
package assignments;

import java.util.*;

public class firstjava {
    // use WHILE loops when UDK the limit of loop
    // in  DO WHILE  min . ek baar chalega loop before checking for conditions
    // sc.input().trim().charAt().......trim() will remove everything extra from aage peeche of the string lik
    // charAt(0)....charater at 0 pos
    // SHORTCUT: fori-> for loop ke liye
    // PALINDROME karne ka naya tareeka !!!!!
    public static void main(String[] args) {

// 1. WAP to check odd or even


//        System.out.print("ENTER YOUR NUMBER : ");
        Scanner sc=new Scanner(System.in);
//        int num=sc.nextInt();
//        if(num%2==0){
//            System.out.println("EVEN number");}
//        else {
//            System.out.println("ODD nummber");
//        }


// 2. WAP greetings for name
//        System.out.print("ENTER UR NAME : ");
//        String name=sc.nextLine();
//        System.out.println("GREETING!!..." + name);
// 3. WAP cal. simple interest
//        System.out.print("ENTER PRINCIPLE amount : ");
//        int prcpl=sc.nextInt();
//        System.out.print("ENTER the TIME period (in years) : ");
//        int tim=sc.nextInt();
//        System.out.print("ENTER the RATE of Interest (%) : ");
//        int rate=sc.nextInt();
//
//        int si= (prcpl*tim*rate)/100;
//        System.out.println("THE SIMPLE INTEREST IS = "+ si);


// 4. WAP calulator
//        System.out.print("ENTER FIRST NUMBER : ");
//        int num1= sc.nextInt();
//        System.out.print("ENTER SECOND NUMBER : ");
//        int num2= sc.nextInt();
//
//        System.out.print("ENTER Function (+,/,*,-) : ");
//        String fncn= sc.next();
//
//        if (fncn.equals("+")){
//            System.out.println("SUM is = "+ (num1+num2));
//        }
//        if (fncn.equals("-")){
//            System.out.println("Difference is = "+ (num1-num2));
//        }
//        if (fncn.equals("/")){
//            System.out.println("Division is = "+ (num1/num2));
//        }
//        if (fncn.equals("*")){
//            System.out.println("Product is = "+ (num1*num2));
//        }

// 5. WAP Largest of 2
//        System.out.print("first num = ");
//        int i =sc.nextInt();
//        System.out.print("second num = ");
//        int j = sc.nextInt();
//        if (i<j){
//            System.out.println(j + "is greater");
//        }else{
//            System.out.println(i + "is greater");
//        }

// 6. WAP for currency coversion from inr to usd
//        System.out.println("enter indian money ");
//        int mon= sc.nextInt();
//        System.out.println("usd =" + mon*0.012);;


// 7. WAP for fibonacci series

//        System.out.println("enter number till itereation : ");
//        int num= sc.nextInt();
//        int a=0,b=1,i=1;
//        System.out.print(a +",");
//        System.out.print(b +",");
//        while (i<num) {
//            i+=2;
//
//            int c=a+b;
//            System.out.print(c +",");
//            a=b;
//            b=c;
//
//
//        }

// 8. WAP for string palindrome

//        System.out.println("ENTER STRING : ");
//        String str=sc.next();
//
//        for (int i = 0, j = str.length() - 1; i < j; i++, j--) {
//            if (!str.charAt(i).equals(str.charAt(j))){
//                System.out.println("not palindrome");
//            }else{
//                System.out.println("palindrome");
//            }

        }








    }
```

Here is the review of your code: **Marks scored: 4/10**

Pros: - Variables are named which is good for readability. - Command input is being used for interactive programming.

Cons (with marks deducted and reason): - Improper use of comments (-1). Comments should be used to explain complex parts of the code, not to replace proper code documentation. - Poor file naming convention (-2). Classes should be PascalCase, not lowercase. - Lots of code is commented out, which makes it difficult to follow what the student is trying to do (-1). - No use of packages or code segregation through methods or classes (-2). This makes the code hard to maintain, dissect and read.

recursionSubset_Self.java

Your code was:

```
package assignments;

import org.w3c.dom.ls.LSOutput;

import java.util.ArrayList;
import java.util.Arrays;

public class recursionSubset_Self {
    public static void main(String[] args) {
//        System.out.println(skip_apple("bacapplecadah"));
//        System.out.println(prmtations_arrlist("","abc"));
        dice("",4);    }
```

```java
// 1. SKIP

    static void skip_a(String p, String up){
        if(up.isEmpty()){
            System.out.println(p);
            return;
        }

        char ch = up.charAt(0);

        if (ch=='a'){
            skip_a(p,up.substring(1));
        }else{
            skip_a(p+ch,up.substring(1));
        }

    }

    static String skip_apple(String up){
        if(up.isEmpty()){
            return "";
        }


        if (up.startsWith("apple")){
            return skip_apple(up.substring(5));
        }else{
            return up.charAt(0)+skip_apple(up.substring(1));
        }

    }
    static int prmtations(String p,String up){
        if (up.isEmpty()){
            System.out.println(p);
            return 1;
        }
        int count =0;

        char ch = up.charAt(0);

        for (int i = 0; i <= p.length(); i++) {
            count=count +prmtations(p.substring(0,i)+ch+p.substring(i,p.length()),up.substring(1));
        }
        return count;
    }
    static void cnt(int n){
        System.out.println(n);
    }

    static ArrayList<String> prmtations_arrlist(String p ,String up){

        ArrayList<String> arr=new ArrayList<>();
        if (up.isEmpty()){
            arr.add(p);
            return arr;
        }

        char ch = up.charAt(0);
        ArrayList<String> ans=new ArrayList<>();
        for (int i = 0; i <= p.length() ; i++) {
            ans.addAll(prmtations_arrlist(p.substring(0,i)+ch+p.substring(i,p.length()),up.substring(1)));
        }
        return ans;
    }
    static void nokia(String p, String up){
        if (up.isEmpty()){
            System.out.println(p);
            return;
        }

        int ch= up.charAt(0) - '0';          // CONVERTS '2' TO 2

        if (ch>=2 && ch<=6){
            for (int i = 3*(ch-2); i < 3*(ch-1); i++) {

                nokia(p+(char)('a'+i),up.substring(1));
            }
        }else if (ch==7){
            for (int i = 15; i < 19; i++) {
                nokia(p+(char)('a'+i),up.substring(1));
            }
        } else if (ch==8) {
            for (int i = 19; i < 22; i++) {
                nokia(p+(char)('a'+i),up.substring(1));

            }
        } else if (ch==9) {
            for (int i = 22; i < 26; i++) {
                nokia(p +(char) ('a' + i), up.substring(1));

            }
        } else {
            nokia(p + '0', up.substring(1));
        }




    }

    static ArrayList<String> nokia_list(String p , String up){
        ArrayList<String> arr=new ArrayList<>();
        if (up.isEmpty()){
            arr.add(p);
            return arr;
        }
        ArrayList<String> ans = new ArrayList<>();
        int ch= up.charAt(0) - '0';          // CONVERTS '2' TO 2

        if (ch>=2 && ch<=6){
            for (int i = 3*(ch-2); i < 3*(ch-1); i++) {

                ans.addAll(nokia_list(p+(char)('a'+i),up.substring(1)));
            }
        }else if (ch==7){
            for (int i = 15; i < 19; i++) {
                ans.addAll(nokia_list(p+(char)('a'+i),up.substring(1)));
            }
        } else if (ch==8) {
            for (int i = 19; i < 22; i++) {
                ans.addAll(nokia_list(p+(char)('a'+i),up.substring(1)));

            }
        } else if (ch==9) {
            for (int i = 22; i < 26; i++) {
                ans.addAll(nokia_list(p +(char) ('a' + i), up.substring(1)));

            }
        } else {
            ans.addAll(nokia_list(p + '0', up.substring(1)));
        }
        return ans;

    }

    static void dice(String p, int tgt){
        if (tgt==0){
            System.out.println(p);
            return;
        }
        for (int i = 1; i <=6 && i<=tgt; i++) {
            dice(p+i,tgt-i);
        }
    }


}
```

Here is the review of your code: **Marks scored: 7/10**

Pros: - The code is neat and easy to read. - Good use of String method `substring` to emulate a recursive step. - Good implementation of recursion in methods such as `skip_a`, `skip_apple`, `prmtations`, `nokia`, among others.

Cons (with marks deducted and reason): - The filename 'recursionSubset_Self.java' is not up to Java standard naming conventions (-1 mark). It should start with an uppercase and be a descriptive noun phrase, e.g. 'RecursionSubset.java'. - The imported `org.w3c.dom.ls.LSOutput` is not used in the code (-1 mark). - Lack of method and variable documentation/comments that could make it easier for other developers to understand (-1 mark). - Function names such as `cnt`, `prmtations_arrlist`, `nokia_list` do not follow camelCase convention used in Java (-0.5 mark). - More descriptive variable and function names could be used for better clarity (-0.5 mark). For instance, `p` and `up` are unclear in their intention.

binarysearch.java

Your code was:

```
package assignments;

import java.util.*;
public class binarysearch {
    public static void main(String[] args) {


//  GAME QUESTION (easy wala)

//        Scanner sc=new Scanner(System.in);
//        System.out.print("ENTER THE RANGE OF THE GAME = ");
//        int ran=sc.nextInt();
//        System.out.print("ENTER FIRST BAD TERM");
//        int bad= sc.nextInt();
//        int start=0;
//        int end=ran-1;
//          badversion(ran,bad,start, end);
////
//          while(start<=end){
//              int cent=mid(start+((end-start)/2));
//              System.out.println("is ur number "+arr[cent]+"(H(higher) / L(lower) / E(equal))");
//              String anser=sc.nextLine();
//              if(anser.charAt(0)=='E'||anser.charAt(0)=='e'){
//                  System.out.println("UR NUMBER IS "+ cent);
//              } else if (anser.charAt(0)=='L'||anser.charAt(0)=='l') {
//                  end=cent-1;
//              }else{
//                  start=cent+1;
//              }

//          }


// 1. BAD VERSION(easy)


//          Scanner sc=new Scanner(System.in);
//          System.out.print("ENTER THE RANGE OF THE GAME = ");
//          int ran=sc.nextInt();
//          System.out.print("ENTER FIRST BAD TERM =");
//          int bad= sc.nextInt();
//          int[] arr=new int[ran];
//
//          // create array
//          for (int i = 0; i < ran; i++) {
//              arr[i]=i+1;
//          }
//
//          //searching
//          int start=0;
//          int end = ran -1;
//
//          while(start<=end){
//              int mid=start+(end-start)/2;
//
////              //middle index
////              if (end%2==0){
////                  mid=(end/2)+1;
////              }else{
////                  mid=(end+1)/2;
////              }
//
//              if(arr[mid]<bad) {
//                  start=mid+1;
//              } else if (arr[mid]>bad) {
//                  end=mid-1;
//              }else{
//                  System.out.print(arr[mid]+" is the first bad");
//                  break;
//              }
//          }


// 2. FIRST AND LAST POSITION (medium)
//          int[] arr = {5,7,7,8,8,10};
//          int bad=8;
//
//
//          //searching
//          int start=0;
//          int end = arr.length -1;
//          boolean check=false;
//
//          while(start<=end){
////              int mid=start+(end-start)/2;
//              int mid=0;
//              if (end%2==0){
//                  mid=(end/2)+1;
//              }else{
//                  mid=(end+1)/2;
//              }
//
//              if(arr[mid]<bad) {
//                  start=mid+1;
//              } else if (arr[mid]>bad) {
//                  end=mid-1;
//              }else{
//                  for (int i = mid;i<end+1 ; i++) {
//                      if (arr[i]==bad){
//                          System.out.print(i+" ");
//                      }
//                  }
//                  check=true;
//                  break;
//              }
//
//
//          }
//          if (check==false){
//              System.out.println(-1);
//          }


// 3. TWO sum II


//          int [] numbers = {-1,0};
//          int [] ans=new int[2];
//          int target=-1;
//
//          for (int i = 0; i < numbers.length; i++) {
//              for (int j = 1; j < numbers.length; j++) {
//
//                  if (numbers[i]!=numbers[j]){
//                      if(numbers[i]+numbers[j]==target){
//                          ans[0]= i+1;
//                          ans[1]= j+1;
//                      }
//
//                  }
//
//              }
//          }
//          System.out.print(Arrays.toString(ans));
//         int[] num1={1,2,3,0,0,0};
//         int m=3;
//         int[] num2={2,5,6};
//         int n=3;
//
//         for (int i = m, j=0; i < num1.length && j< num2.length; i++,j++) {
//             num1[i]=num2[j];
//         }
//
//         // bubble sort
////         for (int i = 1; i < num1.length; i++) {
////             if(num1[i]<num1[i-1]){
////                 int temp=num1[i];
////                 num1[i]=num1[i-1];
////                 num1[i-1]=temp;
////             }
////         }
////         System.out.println(Arrays.toString(num1));
//
//         // cyclic sort
//         int i=0;
//         while(i< num1.length){
//             int corct=num1[i]-1;
//             if(num1[i]!=corct){
//                 int temp=num1[i];
//                 num1[i]=num1[corct];
//                 num1[corct]=temp;
//             }else{
//                 i++;
//             }
//         }
//         System.out.println(Arrays.toString(num1));
//
//
//     }
//     static void badversion(int range,int bad, int start, int end){
//         //create array
//         int[] arr=createarr(range);
//         binary(start,end,arr,bad);
//
//
//     }
//
//     static void binary(int start, int end, int[] array, int tgt){
//         while (start<=end){
//             int cent=mid(end);
//             if (array[cent]==tgt) {
//                 System.out.println(array[cent]+" is the first bad");
//             } else if (array[cent]<tgt) {
//                 start=cent +1;
//             }else{
//                 end=cent-1;
//             }
//
//         }
//     }
//
//     static int mid(int num){
```

```
//        if (num%2==0){
//            return (num/2)-1;
//        }else{
//            return ((num+1)/2)-1;
//        }
//    }
//    static int[] createarr(int limit){
//        int[] sample = new int[limit];
//        for (int i = 0; i < limit; i++) {
//            sample[i]=i+1;
//        }
//        return sample;
//
//    }
    }}
```

Here is the review of your code: **Marks scored: 5/10**

Pros: - The code segments seem to include diverse problem-solving approaches such as binary search, sorting, etc. - Some code comments are written and can aid understanding. - Use of standard Scanner class for user inputs.

Cons (with marks deducted and reason): - Inappropriate file name 'binarysearch.java'. It should be named as 'BinarySearch.java' adhering to the Java naming conventions. **(-0.5 Marks)** - Multiple code segments commented out, which makes the code understanding, debugging, and grading difficult. **(-0.5 Marks)** - The code is cluttered and lacks proper structure and indentation. Proper use of whitespaces and indentation is lacking throughout. **(-1 Mark)** - The variable and method names are not descriptive and do not adhere to Java naming conventions. Eg: 'ran', 'bad', 'start', 'cent' are not meaningful. **(-1 Mark)** - Redundant code and repetitive codes which could have been modularized. Code reusability and encapsulation is not implemented. **(-1 Mark)** - No use of access modifiers, all the variables and methods are using the default access modifier. **(-0.5 Marks)** - No exception handling. **(-0.5 Marks)**

math.java

Your code was:

```
package assignments;

import java.util.*;


public class math {
    public static void main(String[] args) {
        String s="IX";
        System.out.println(romanToInt(s));
//        int n=4;
//        System.out.println('AA'-'A');
    }
    static int romanToInt(String num){
        int sum=0;
        Character[] roman={'I','V','X','L','C','D','M'};
        int[] val={1,5,10,50,100,500,1000};
        int [] savedval= new int[num.length()];
        for (int i = 0; i < num.length(); i++) {
            char romanCh=num.charAt(i);
            savedval[i]=val[binary(0,roman.length-1,roman,romanCh)];
        }
        for (int i = 1; i <= savedval.length; i++) {
            if(i== savedval.length){
                sum+=savedval[i-1];
                break;
            }
            if (savedval[i-1]<savedval[i]){
                sum-=savedval[i-1];
            }else{
                sum+=savedval[i-1];
            }
        }
        return sum;
    }
    static int binary(int start, int end, Character[] array, char tgt){
        while (start<=end){
            int mid= start+(end-start)/2;
            if (array[mid]==tgt) {
                return mid;
            } else if (array[mid]<tgt) {
                start=mid +1;
            }else{
                end=mid-1;
            }

        }
        System.out.println("wrong roman "+ tgt);
        return -1;

    }
//    static int romanToInt_betterone(String num) {                                   // BETTER CODE AND A
//        int[] values = new int[26]; // Assuming only uppercase Roman numerals
//        values['I' - 'A'] = 1;
//        values['V' - 'A'] = 5;
//        values['X' - 'A'] = 10;
//        values['L' - 'A'] = 50;
//        values['C' - 'A'] = 100;
//        values['D' - 'A'] = 500;
//        values['M' - 'A'] = 1000;
//
//        int sum = 0;
//        int prevValue = 0;
//
//        for (int i = num.length() - 1; i >= 0; i--) {
//            int value = values[num.charAt(i) - 'A'];
//            if (value < prevValue) {
//                sum -= value;
//            } else {
//                sum += value;
//            }
//            prevValue = value;
//        }
//        return sum;
//    }
//    static int romanToInt_bettercode(String num) {                                 // BETTER CODE
//        int[] val = {1, 5, 10, 50, 100, 500, 1000};
//        int sum = 0;
//        int prevValue = 0;
//
//        for (int i = num.length() - 1; i >= 0; i--) {
//            char romanCh = num.charAt(i);
//            int index = indexOf(romanCh);
//            if (val[index] < prevValue) {
//                sum -= val[index];
//            } else {
//                sum += val[index];
//            }
//            prevValue = val[index];
//        }
//
//        return sum;
//    }
//
//    static int indexOf(char romanCh) {
//        switch (romanCh) {
//            case 'I': return 0;
//            case 'V': return 1;
//            case 'X': return 2;
//            case 'L': return 3;
//            case 'C': return 4;
//            case 'D': return 5;
//            case 'M': return 6;
//            default: throw new IllegalArgumentException("Invalid Roman numeral character: " + romanCh);
//        }
//    }
//}
   static boolean happyNum(int num){
       int tgt=sum(0,num);
       if (tgt/10<10){
           if(tgt==1){
               return true;
           }
       }else{
           happyNum(tgt);
       }
       return false;
   }
   static int sum(int ans,int num){

        if (num==0){
            return ans ;
        }
        return sum(ans=((num%10)*(num%10)),num/10);


   }

   static String armstrong(int num){
        if (cube(0,num)==num){
            return "yes";
        }
        return "no";
   }
   static int cube(int ans, int num){
        if (num==0){
            return ans;
        }
        return cube(ans+=((num%10)*(num%10)*(num%10)),num/10);
    }

    static boolean powOf4(float num){
        if(quad(num)==4.0){
            return true;
        }
        return false;
    }
    static float quad(float num){
        if (num>0 && num<10){
            return num;
        }
        return quad(num/4);
    }
    static int factorial(int num){
        int sum=1;
        while(num>=1){
            sum=sum*num;
            num--;
        }
        return sum;
    }
//    static String excel(int num){
//        int[] reference=new int[(int) (Math.pow(2,31)-1)];
//    }



}
```

Here is the review of your code: **Marks scored: 5/10**

Pros: 1. Good attempt to solve multiple math problems. 2. Use of `binary` function to find position of a character is appreciated. 3. Recursive methods used to solve problems are logically sound.

Cons (with marks deducted and reason): 1. Filenaming: Though it does not affect the code execution, Java class names should start with an uppercase letter (java naming convention). This is not followed here. `math.java` should be `Math.java`. Marks deducted: 1. 2. Usage of 'single letter variable names' makes the code hard to read and understand. Multiple variables are single character variables (e.g. 's', 'n'). Marks deducted: 1. 3. Code readability: Commented code should be avoided for cleaner code and better readability. Marks deducted: 1. 4. Code efficiency: The implementation of `romanToInt` is inefficient. Doing linear search for finding Roman numeral is inefficient and time-consuming compared to direct mapping. Marks deducted: 1. 5. Code completeness: There's a function `happyNum` that never actually gets called. This function also fails to return any value in certain situations. Marks deducted: 1.

arrays.java

Your code was:

```
package assignments;

import java.util.ArrayList;
import java.util.*;

public class arrays {
    // for (int num : arr).....sout (num  + " ")
    /*this will go on each element of array and print it*/

    // PRIMITIVES :- qntities that cn not be further broken down like boolean, int etc.
    //the are stored in stack
    // OBJECTS :-
    // Stored in heap mem and accessed reference variable (arr[0]).

    // int [][] arr=new int[3][] .......(rows is mandatory).......[array of assignments.arrays]

    // arr[rows][colums]
    // arr[rows].length
    // ENHANCED FOR LOOP for printing
    /* for (int a : arr){
     * sout (Arrays.toString(a))
     * }*/
    // ArrayList<integer> list = new Arraylist <>();
    // for assignments.arrays who u do not know the size
    // list.add() , list.set(index, towhat), list.remove(index), list.get(index)
    // arraylist does have a limit....but when limit is reached....new list created....old ones added to new..

    // CONVERT STRING TO ARRAY.........Arrays.toString(strvariablename.toCharArray())

    public static void main(String[] args) {

//        Scanner sc =new Scanner(System.in);
//        int [] arr = new int[4];
//        int [] arr1 = new int[4];
//
//        for (int i = 0; i < arr.length; i++) {
//            arr[i]=sc.nextInt();
//        }


// 2. WAP for concatenation

/*
        int [] arr2 = new int[arr1.length+arr.length];


        for (int i = 0; i < arr.length; i++) {
            arr[i]=sc.nextInt();
        }
        for (int i = 0; i < arr.length; i++) {
            arr1[i]=sc.nextInt();
        }

        for (int i = 0; i < (arr.length+arr1.length); i++) {
            if(i<arr.length){
                arr2[i]=arr[i];
            }else{
                arr2[i]=arr1[i- arr1.length];
            }


        }
        for (int i = 0; i < arr2.length; i++) {
            System.out.print(arr2[i] + " , ");

        }*/

// 3. sum of elements

/*        int sum=0;

        for (int i = 0; i < arr1.length; i++) {
            for (int j = 0; j <= i; j++) {
                sum=sum+arr[j];

            }
        arr1[i]=sum;
        sum=0;

        }
        for (int i = 0; i < arr1.length; i++) {
            System.out.print(arr1[i] + " , ");
        }*/
```

```
// 4. WAP max val of customer

/*        int[][] accounts={{1,5},{7,3},{3,5}};
        int[] data=new int[accounts.length];
        int sum=0;

        for (int cstmer = 0; cstmer < accounts.length; cstmer++) {
            for (int accbal = 0; accbal < accounts[cstmer].length; accbal++) {
                sum=sum +accounts[cstmer][accbal];
            }
            data[cstmer]=sum;
            sum=0;

        }
        int max=0;

        for (int cstmer = 0; cstmer < data.length ; cstmer++) {

            System.out.println("Customer"+ (cstmer+1)+" = "+ data[cstmer]);
            if (data[cstmer]>max){
                max=data[cstmer];
            }else{
                max=data[cstmer];
            }

        }
        System.out.println("Max value is "+ max);*/

// 5. WAP to shuffle array
/*

        int[] nums = {2,5,1,3,4,7};
        int n=3;
        int[] arr=new int[2*n];


        for (int i = 0; i < n; i++) {
            arr[i*2]=nums[i];
            arr[i*2+1]=nums[i+n];
        }
        System.out.println(Arrays.toString(arr));*/

// 6. KIDS and candies wala

/*        int[] candies = {4,2,1,1,2};
        int extraCandies = 1;
        Boolean[] arr = new Boolean[candies.length];
        int sum=0;

        for (int i = 0; i < candies.length; i++) {
            int temp = candies[i]+extraCandies;


            for (int j = 0; j < candies.length; j++) {
                if (temp >= candies[j]) {
                    sum++;

                }
            }
            if(sum==candies.length){
                arr[i]=true;
            }else{
                arr[i]=false;
            }
            sum=0;
        }
        System.out.print(Arrays.toString(arr));*/


//  23. LUCKY NUMBERS
/*
        int[][] matrix={{1,10,4,2},{9,3,8,7},{15,16,17,12}};
        int minval=0,index=0,max=0;
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 1; j < matrix[i].length; j++) {
                minval=matrix[i][0];

                if (matrix[i][j]<minval){
                    minval=matrix[i][j];
                    index=j;
                }

            }

            for (int j = 0; j < matrix.length; j++) {
                max=matrix[0][index];
                for (int k = 1; k < matrix.length; k++) {
                    if (max < matrix[k][index]) {
                        max = matrix[k][index];
                    }
                }
            if (max==minval){
                System.out.println("luncky nummber = "+ minval);
                break;
                }

            }




        }*/


// 7. GOOD PAIRS IN array
/*
        int[] nums={1,2,3};
        int sum=0;
        for (int i = 0; i < nums.length; i++) {
            for (int j = i+1;j < nums.length; j++) {
                if (nums[i]==nums[j]) {
                    System.out.print("(" + i + "," + j + ")");
                    sum++;
                }
            }
        }
        System.out.println();
        System.out.println("total good pairs = "+sum);*/
```

```
// 8. numbers smaller than current
/*        int[] nums = {7,7,7,7};
        int[] arr=new int[nums.length];
        int grter=0;

        for (int i = 0; i < nums.length; i++) {
            for (int j = 0; j < nums.length; j++) {
                if (nums[i]>nums[j]){
                    grter++;
                }
            }
            arr[i]=grter;
            grter=0;
        }
        System.out.println(Arrays.toString(arr));*/


//        String alpha = "qwertyuiopasdfghjklzxcvbnm";
//        String arr= Arrays.toString(alpha.toCharArray());
//        String word="thequickbrownfoxjumpsoverthelazydog";
//        String check= Arrays.toString(alpha.toCharArray());
//
//        for (int i = 0; i < arr.length(); i++) {
//            for (int j = 0; j < check.length(); j++) {
//                if (arr[i]==check[j]){
//
//                }
//
//            }
//
//        }

//10. PANGRAM (RETRY)


//attempt 1

//        String alpha = "qwertyuiopasdfghjklzxcvbnm";
//        String word="thequickbrownfoxjumpsoverthelazydog";
//        int ran=0;
//
//        for (int i = 0; i < word.length(); i++) {
//            for (int j = 0; j < alpha.length(); j++) {
//                if (word.charAt(i)==alpha.charAt(j)){
//                    ran++;
//                }
//
//            }
//
//        }

//attempt 2

        ////            if (i==0){
////                alpha[i]=sentence.charAt(i);
////            }else{
////                for (int j = 0; j <= i; j++) {
////                    if (alpha[j]!=sentence.charAt(i)) {
////                        alpha[i] = sentence.charAt(i);
////                        break;
////                    }
////                }
////            }
////            System.out.print(alpha[i]);


//attempt 3

//        String sentence = "thequickbrownfoxjumpsoverthelazydog";
//        boolean[] alpha= new boolean[26];
//        boolean pangram=true;
//
//        for (int i = 0; i < sentence.length(); i++) {

//            char ch=Character.toLowerCase(sentence.charAt(i));   // convert to lower case
//            if (Character.isLetter(ch)){                         // check ki letter hai ki nahi
//                alpha[ch-'a']=true;                              // mark wo index true....(a - a=0) , (b - a
//            }
//        }
//        for (int i = 0; i < 26; i++) {
//            if (alpha[i] != true) {
//                pangram = false;
//            }
//
//        }
//
//        if (pangram) {
//            System.out.println("pangram");
//        } else {
//            System.out.println("not pangram");
//        }



// 11. Matching a rule

//        String[][] items = {{"phone","blue","pixel"},{"computer","silver","phone"},{"phone","gold","iphone"}
//        String[] rule={"type","color","name"};
//        String ruleKey = "type";
//        String ruleValue = "phone";
//        int sum=0;
//        for (int i = 0; i < rule.length; i++) {
//
//            if (ruleKey==rule[i]){
//                for (int j = 0; j < items.length; j++) {
//                    if (items[j][i]==ruleValue){
//                        sum++;
//                    }
//                }
//            }
//
//        }
//        if (sum==0){
//            System.out.println("ITEM DOES NOT EXIST");
//        }else{
//            System.out.println(sum);
//        }

// 12. Altitude
//          int[] gain = {-4,-3,-2,-1,4,3,2};
//          int[] alt =new int[gain.length+1];
//          int sum=0;
//
//          for (int i = 0; i < alt.length; i++) {
//              if(i==0){
//                 alt[i]=0;
//              }else{
//                  sum+=gain[i-1];
//                  alt[i]=sum;
//              }
//
//          }
//          int max=alt[1];
//          for (int i = 0; i < alt.length; i++) {
//              if(alt[i]>max){
//                  max=alt[i];
//              }
//          }
//
//          System.out.println(max);


// 13. FLIPPING AN ARRAY


//          int[][] image = {{1,1,0,0},{1,0,0,1},{0,1,1,1},{1,0,1,0}};
//          int[][] result= new int[image.length][image[0].length];
//          int index=0;
//
//          for (int row = 0; row < image.length; row++) {
//              for (int col = image[row].length-1; col >= 0; col--) {
//                  if (image[row][col]==0){
//                      result[row][index]=1;
//                      index++;
//                  }else{
//                      result[row][index]=0;
//                      index++;
//                  }
//              }
//          index=0;
//
//          }
//
//          for (int[] row : result) {                                                          // NEW WAY TO PRINT 2D ARRAY
//              System.out.println(Arrays.toString(row));
//          }


// 14. Cells with odd values..........(RETRY)

//          int mrow=2;
//          int ncol=2;
//          int[][] indices = {{1,1},{0,0}};
//          int[][] result= new int[mrow][ncol];
//
//          for (int row = 0; row < mrow; row++) {
//              int incrow=indices[row][0];
//              int incol=indices[row][1];
//              for (int col = 0; col < ncol; col++) {
```

```
//                  result[row][col]=0;
////                  if(incrow==row){
////                      result[incrow][col]++;
////                  }
////                  if (incol==col){
////                      result[row][incol]++;
////                  }
//                  result[incrow][col]++;
//                  result[row][incol]++;
//
//              }
//
//          }
//          for (int[] row : result) {                                              // NEW WAY TO PRINT 2D ARRAY
//              System.out.println(Arrays.toString(row));}


// 15. MATRIX DIAGONAL SUM

//          int[][] mat = {{1,1,1,1},
//                         {1,1,1,1},
//                         {1,1,1,1},
//                         {1,1,1,1}};
//          int sum=0;
//          for (int row = 0; row < mat.length; row++) {
//              for (int col = 0; col < mat[row].length; col++) {
//                  if (row==col){
//                      sum=sum+mat[row][col];
//                  }
//              }
//              for (int col = mat[row].length-1; col >=0 ; col--) {
//                  if (row==col){
//                      sum=sum+mat[row][col];
//                  }
//
//              }
//
//          }
//          if(mat.length%2!=0){
//              System.out.println(sum-mat[((mat.length+1)/2)-1][((mat.length+1)/2)-1]);
//          }else{
//              System.out.println(sum);
//          }


// 16. EVEN NUMBER OF DIGITS

//          int[] nums = {555,901,482,1771};
//          int sum=0;
//          int even=0;
//          for (int i = 0; i < nums.length; i++) {
//              int j=nums[i]; //pick kia
//              while(j>0){  //no of digits aagaye
//                  j=j/10;
//                  sum++;
//              }
//              if(sum%2==0){
//                  even++;
//              }
//              sum=0;
```

```
//
//        }
//        System.out.println(even);

// 17. TRANSPOSE MATRIX


//        int[][] matrix = {{1,2,3},{4,5,6},{7,8,9}};
//        int[][] result= new int[matrix[0].length][matrix.length];
//        for (int row = 0; row < matrix.length; row++) {
//            for (int col = 0; col < matrix[row].length; col++) {
//                result[col][row]=matrix[row][col];
//            }
//        }
//        for (int i = 0; i < result.length; i++) {
//            System.out.println(Arrays.toString(result[i]));
//        }









    }
}
```

Here is the review of your code: **Marks scored: 4/10**

Pros: - Varied examples and problems covered. - The code makes use of loops, arrays and array manipulation techniques properly.

Cons (with marks deducted and reason): - Deducted 2 points because the code is not properly organized. All the code is written inside the main method and no other methods are used. This makes the code hard to read and lacks use of encapsulation. - Deducted 2 points for a lack of error handling. The code does not include any try/catch blocks or throws any exceptions. - Deducted 1 point for the absence of comments in several parts of the script makes it hard for others to understand the functionality of several code blocks. - Deducted 1 point for poor naming conventions. The filename 'arrays.java' doesn't follow the standard Java convention of each word in camel case starting with a capital letter.

sudokuSolver.java

Your code was:

```
package assignments;

public class sudokuSolver {
    public static void main(String[] args) {
        int[][] board=new int[][]{{5, 3, 0, 0, 7, 0, 0, 0, 0},
                {6, 0, 0, 1, 9, 5, 0, 0, 0},
                {0, 9, 8, 0, 0, 0, 0, 6, 0},
                {8, 0, 0, 0, 6, 0, 0, 0, 3},
                {4, 0, 0, 8, 0, 3, 0, 0, 1},
                {7, 0, 0, 0, 2, 0, 0, 0, 6},
                {0, 6, 0, 0, 0, 0, 2, 8, 0},
                {0, 0, 0, 4, 1, 9, 0, 0, 5},
                {0, 0, 0, 0, 8, 0, 0, 7, 9}};

        if (solve(board)){
            display(board);
        }else{
            System.out.println("no");
        }
    }
    static boolean solve(int[][] board){
        int n= board.length;
        int col= -1;
        int row= -1;
        boolean emptySpace=false;

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if(board[i][j]==0){
                    row=i;
                    col=j;
                    emptySpace=true;
                    break;
                }
            }
            if(emptySpace==true){
                break;
            }

        }
        if (emptySpace==false){
            return true;
        }

        for (int i = 1; i <= n; i++) {
            if (isSafe(board,row,col,i)){
                board[row][col]=i;
                if(solve(board)){
                    return true;
                }else{
                    board[row][col]=0;
                }
            }
        }

        return false;
    }
    static void display(int[][] board){
        for(int[]row:board){
            for (int ele:row){
                System.out.print(ele+" ");
            }
            System.out.println();
        }
    }



    static boolean isSafe(int[][] board, int row, int col, int num){
        //check row
        for (int i = 0; i < board.length; i++) {
            if(board[row][i]==num){
                return false;
            }
        }

        //check col
        for (int i = 0; i < board.length; i++) {
            if(board[i][col]==num){
                return false;
            }
        }

        //check box
        int sqrt=(int)(Math.sqrt(board.length));
        int starRow = row - row % sqrt;
        int startCol= col - col % sqrt;

        for (int i = starRow; i <starRow+sqrt ; i++) {
            for (int j = startCol; j < startCol+sqrt; j++) {
                if(board[i][j]==num){
                    return false;
                }
            }

        }
        return true;

    }
}
```

Here is the review of your code: **Marks scored: 8/10**

Pros: - The code is easy to understand, with clear variable names. - The code executes correctly and provides the correct output for the sudoku problem. - Excellent use of recursion in the solve method.

Cons (with marks deducted and reason):

- Deducted 1 mark for filename. In Java, filenames should match the class name and start with a capital letter, i.e., SudokuSolver.java, not sudokuSolver.java.
- Deducted 1 mark for class naming. As per Java conventions, the class name 'sudokuSolver' should be in Pascal Case like 'SudokuSolver'.
- No comments in the code explaining what certain functions do, which makes it harder for others to understand the approach.

Media.java

Your code was:

```
package OOPs.video5.car;

public interface Media {
    void start();
    void stop();
//    farak nahi pada qki stop already hai kisi aur class mei

}
```

Here is the review of your code: **Marks scored: 7/10**

Pros: - The filename is correctly named as per the class name.

Cons (with marks deducted and reason): - Marks deducted: 1. The code includes comments which are not in English. It's better to use English for global understanding. - Marks deducted: 1. The interface is not fully utilizing its feature. More methods could have been added to make it a good contract. - Marks deducted: 1. No Java Doc comments included. It's better to provide information about what the class or methods do using Java Doc comments.

PowerEngine.java

Your code was:

```
package OOPs.video5.car;

public class PowerEngine implements Engine{
    @Override
    public void start() {
        System.out.println("power chala");
    }

    @Override
    public void stop() {
        System.out.println("power ruka");
    }

    @Override
    public void acc() {
        System.out.println("power baga");
    }
}
```

Here is the review of your code: **Marks scored: 7/10**

Pros: - The filename (PowerEngine.java) adheres to Java conventions. - The code is simple and easy to understand. - The code follows Java coding standards and best practices.

Cons (with marks deducted and reason): - Minus 1 mark for not providing class-level and method-level comments. - Minus 2 marks because the messages printed in the methods are not in English, hampering understandability and universality.

Engine.java

Your code was:

```
package OOPs.video5.car;

public interface Engine {
    //by default static and final hai isliye grey
    static final int price = 2220000;
    void start();
    void stop();
    void acc();
}
```

Here is the review of your code: **Marks scored: 8/10**

Pros: - The code is quite straightforward and easy to understand. - The filename `Engine.java` is appropriate as it aligns with the interface name and Java naming conventions. - The interface definition and its method signatures are defined correctly.

Cons (with marks deducted and reason): - Deducted 1 point since a good practice is to include a brief comment explaining the purpose of the interface. - Deducted 1 point for the non-English comment in the code. Comments should be in English and clearly explain the code.

Main.java

Your code was:

```
package OOPs.video5.car;

public class Main {
    public static void main(String[] args) {
//        Car supra=new Car();
//        supra.acc();
//        supra.brake();
//        supra.stop();

//        System.out.println(Engine.price);
        NiceCar car = new NiceCar();
        car.acc();
        car.startMusic();

        car.engupgrade();
        car.acc();
        car.stop();



    }
}
```

Here is the review of your code: **Marks scored: 7/10**

Pros: - The code syntax is correct and the filename is as per recommended Java conventions. - The code is simple, well-written and easy to understand.

Cons (with marks deducted and reason): - Deduction of 2 points for unused or commented lines of code. These should be removed before submission to maintain code cleanliness. - Deduction of 1 point for not having comments to explain blur code lines which would have made the code more understandable.

Brake.java

Your code was:

```
package OOPs.video5.car;

public interface  Brake {
    void brake();
    void start();

}
```

Here is the review of your code: ** Marks scored: 9.5/10 **

** Pros: ** - The code is neat and follows the Java interface conventions correctly. - The filename matches with the interface name.

** Cons (with marks deducted and reason):** - 0.5 marks were deducted because there's no comments or explanations given for what each method does inside the interface. This makes it difficult for others to understand the purpose of each method.

NiceCar.java

Your code was:

```
package OOPs.video5.car;

public class NiceCar {
    private Engine eng;
    private Media player=new CD();
    //default engine type
    public NiceCar(){
        eng = new PowerEngine();
    }
    public void engupgrade(){
        eng=new ElectricEngine();
    }
    public NiceCar(Engine eng){
        this.eng=eng;
    }
    public void start(){
        eng.start();
    }
    public void stop(){
        eng.stop();
    }
    public void acc(){
        eng.acc();
    }
    public void startMusic(){
        player.start();
    }
    public void stopMusic(){
        player.stop();
    }
}
```

Here is the review of your code: **Marks scored: 8/10**

Pros: - The naming conventions used in the program are in accordance with the Java standards. - The use of encapsulation through private fields is appropriate and OOP concepts are well implemented. - The file name is intuitive and adheres to the naming standard having the same name as the public class.

Cons (with marks deducted and reason): - Deducted 1 point because although declaring Media 'player' as a 'new CD()' at the top of class is not technically wrong, it is more common to place object instantiation within constructor or method for better understanding. - Deducted 1 point because there is no comments to explain methods, which could improve understanding and maintenance of the code.

ElectricEngine.java

Your code was:

```
package OOPs.video5.car;

public class ElectricEngine implements Engine{
    @Override
    public void start() {
        System.out.println("Tesla chali");
    }

    @Override
    public void stop() {
        System.out.println("Tesla ruki");
    }

    @Override
    public void acc() {
        System.out.println("Tesla bhagi");
    }
}
```

Here is the review of your code: **Marks scored: 7/10**

Pros: - The file name ElectricEngine.java adheres to standard conventions. - The code is implementating methods from an Interface follows OOPs best practices. - The code is generally well-written and easily understandable.

Cons (with marks deducted and reason): - 2 points deducted as the output messages are not written in English. It makes the error and output results hard to understand. Message should be universally understandable. - 1 point deducted for the lack of comments. Comments could greatly enhance code understanding.

Car.java

Your code was:

```
package OOPs.video5.car;

public class Car implements Engine,Brake,Media {

    // INTERFACE se multiple inheritance kar sakte hai
    @Override
    public void brake() {
        System.out.println("bbhadiya break");
    }

    @Override
    public void start() {
        System.out.println("makhan start");
    }

    @Override
    public void stop() {
        System.out.println("gaadi band");
    }

    @Override
    public void acc() {
        System.out.println("baga bc");
    }
}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - The filename Car.java adheres to the Java naming conventions. - The usage of interfaces for multiple inheritance is correct and good. - The methods are properly overridden and implemented.

Cons (with marks deducted and reason): - Deducted 2 points because the code does not follow Java conventions and good practices. Method and package names are not clear and indented properly. - Deducted 1 point due to inappropriate print statements. They should be more formal and informative. - Deducted 1 point as there is a lack of comments for understanding the logic behind the code which makes it less readable.

CD.java

Your code was:

```
package OOPs.video5.car;

public class CD implements Media {

    @Override
    public void start() {
        System.out.println("gana chal gaya");
    }

    @Override
    public void stop() {
        System.out.println("gaana band ho gaya");
    }
}
```

Here is the review of your code: **Marks scored: 8/10**

Pros: - The filename "CD.java" is correct as per Java conventions for class names. - The class correctly implements an interface and its methods.

Cons (with marks deducted and reason): - Deducted 1 mark as the student did not provide comments for the methods. Comments improve code readability. - Deducted 1 mark as the student used non-English phrases in printed statements which may not be universally understandable.

Main.java

Your code was:

```java
package OOPs.video5.family;

public class Main {
    public static void main(String[] args) {
        Son son = new Son(20);
//        son.career("Suvrat");
//        son.partner("Aastha", 20);
        int age = son.age;
        System.out.println(age);

        Daughter beti= new Daughter(30);
//        beti.career("minority");
//        beti.partner("harvanvi", 40);


        //qki parent abstract hai isliye override hoga

//        parent mom = new parent() {
//            @Override
//            void career(String name) {
//
//            }
//
//            @Override
//            void partner(String name, int age) {
//
//            }
//        }
        son.greet();

    }
}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - The code is highly readable and easy to understand. - The filename `Main.java` adheres to Java standards for a class. - Usage of informative variable names aids in understanding the purpose of those variables.

Cons (with marks deducted and reason): - Inefficient code usage: Unnecessarily created variables like `age` and `beti` etc (-1 mark). - Lack of comments explaining the purpose of the code blocks (-1 mark). - Indentation and formatting issues exist throughout the code, not adhering to standard Java coding conventions (-1 mark). - Unused code lines, commented out, adds clutter. Code should be clean with only relevant sections (-1 mark).

parent.java

Your code was:

```java
package OOPs.video5.family;

public abstract class parent {
    int age;
    public parent(int age){
        this.age=age;
    }
    abstract void career (String name);
    abstract void partner(String name, int age);

    void greet(){
        System.out.println("hello bkl");
    }
}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - Use of well-defined abstract methods. - Correct use of the `this` keyword. - Use of constructor for initialization of variables.

Cons (with marks deducted and reason): - The class name should start with an uppercase character. Java naming conventions recommend to use camel case for class names ('Parent' instead of 'parent'). -1 mark. - The use of vague variable names such as "bkl" in the `greet` method is unprofessional and decreases readability, -1 mark. - The filename should ideally be 'Parent.java' instead of 'parent.java', adhering to Java file naming conventions. -1 mark. - Lack of usage of access modifiers (private, public, protected) for variables, -1 mark.

Daughter.java

Your code was:

```
package OOPs.video5.family;

public class Daughter extends parent {
    public Daughter(int age) {
        super(age);
    }

    @Override
    void career(String name) {
        System.out.println("ill be a housewife coz im a " + name);
    }

    @Override
    void partner(String name, int age) {
        System.out.println("ill marry " + name + "hes "  + age);
    }
}
```

Here is the review of your code: **Marks scored: 4/10**

Pros: - The code is neatly structured and easy to follow. - Appropriate use of Inheritance and Overriding methods from parent class. - Correctly named the file as 'Daughter.java' as per Java conventions.

Cons (with marks deducted and reason): - Deducted 2 marks because class names in Java should start with a capital letter. 'parent' should be 'Parent'. - Deducted 3 marks for hardcoding responses in 'career' and 'partner' methods. The code lacks flexibility and doesn't follow best practices. - Deducted 1 mark since 'name' and 'age' variables are not validated for correct values. Edge cases are not handled.

Son.java

Your code was:

```
package OOPs.video5.family;

import OOPs.video5.family.parent;

public class Son extends parent {

    @Override
    void career(String name) {
        System.out.println("ill be a rapper coz im " + name);
    }

    @Override
    void partner(String name, int age) {
        System.out.println("ill marry "+ name + "and she is " + age );
    }
    public Son(int age){
        super(age);
    }

}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - The code is simple and clear, demonstrating basic inheritance usage in Java. - The file name 'Son.java' is correctly named, following PascalCase convention.

Cons (with marks deducted and reason): - 1 mark deducted for not following Java naming conventions. Class names should be nouns, in mixed case with the first letter of each internal word capitalized. - 1 mark deducted for not providing method comments/documentation. - 1 mark deducted for the class 'Son' extending 'parent'. Class names should be capitalized ('Parent'). - 1 mark deducted since there seems to be lack of encapsulation, private or protected members are not defined in the parent class and there are no getters and setters.

OOPS.java

Your code was:

```
package OOPs.video1;


class pen{
    String colour;                              //STRUCTURE
    String type;

    public void write(){                        //MEATHOD
        System.out.println("kuch bhi");
    }
    pen(){                                        //normal contructor
        System.out.println("this is contructor");
    }
    pen(String colour, String type){
        this.colour=colour;
        this.type=type;
    }
    public void peninfo(){
        System.out.println(this.colour);
        System.out.println(this.type);
    }

}
public class OOPS {
    public static void main(String[] args) {
//        pen one=new pen();                      //pen()--constructor...only called once
//        one.colour="blue";
//        one.type="gel";
//        one.write();
        pen pen1=new pen("red","ball");
        pen1.peninfo();
    }


    // java has no destructor coz it has garbage collector

}
```

Here is the review of your code: **Marks scored: 6/10**

Pros: - The code is simple and easy to understand. - The use of constructor to initialize the object is good practice. - The method names are self-explanatory and precise.

Cons (with marks deducted and reason): - The class name 'pen' should be in Pascal case as per Java naming conventions. It would be Pen. (-1) - The variable 'colour', 'type' are package-private. It would be better to make them private.(-1) - The file name 'OOPS' is not descriptive and not adhering to standards (-1) - Unnecessary commented code should be removed for readability (-1)

functOver.java

Your code was:

```
package OOPs.video1;

class Student{
    String name;
    int age;
    public void kid(){
        System.out.println("NO args");
        System.out.println(this.name+ " "+ this.age);
    }
    public void kid(String name, int age){
        System.out.println("2 Agruments ");
        System.out.print(name +" ");
        System.out.println(age);
    }
//copy function
    Student(Student s1 ){
        this.name=s1.name;
        this.age=s1.age;
    }

    Student (){

    }

//Function overload.....function name same ho sakte hai par arguments alag hone
    public void std(String name){
        System.out.println(name);
    }
    public void std(int age){
        System.out.println(age);
    }
    public void std(String name,int age){
        System.out.println(name + " "+ age);
    }


}
public class functOver {
    public static void main(String[] args) {
        Student s1=new Student();
        s1.name="pipi";
        s1.age=44;
        s1.kid();
        s1.kid("suvv",22);

        Student s2=new Student();
        System.out.println();
        s2.name="sggfsji";
        s2.age=44;
        s2.kid();
        s2.kid("gddg",22);
        Student s3=new Student(s2);
        System.out.println();
        s3.kid();
        s3.kid("sambar",15);

    }
}
```

Here is the review of your code: ** Marks scored: 5/10 **

** Pros: ** - Use of overloading and copy constructor - Logical flow of the program is fine - The usage of comments to separate sections of code is appreciated.

** Cons (with marks deducted and reason): ** - Deducted 3 pts for not declaring instance variables as private in the Student class in line with Java best practices for encapsulation. Use setters and getters for properties. - Deducted 1 pt for file name standard: Files should match the class name, rename “functOver.java” to “FunctOver.java”. - Deducted 1 pt for inconsistency in code, there are undocumented functions, and non-meaningful function and variable names. Code readability is important.