Computer Project

Aanjishnu Bhattacharyya

August 4, 2021

### Assignment 1

Writa a pogram to take lower and upper range from the user and print all the Kaprekar numbers within that range. (A number n hanving d digits is squared and split into two pieces, right hand piece having d digits and left hand piece having d or d-1 digits. If sum of the two peices is equal to the number then n is Kaprekar number). Eg: 9, 45, [55], 297

### Algorithm:

**Class Kaprekar\_main:**  
**Method Main:**  
Step 1: declare lr and ur as lower and upper limit respectively  
Step 2: accept input from the user the lower and upper limit in lr and ur  
Step 3: check if the numbers are greater than zero and ur ¿ lr. if not then renter  
Step 4: initialize a Kaprekar object.  
Step 5: call the display method of Kaprekar object  
  
**Class Kaprekar:**  
**Method display:**  
Step 1: declare i as a loop control variable   
Step 2: start a ‘for’ loop from lower\_range to upper\_range with i as the loop control.  
Step 3: call iskaprekar method passing i as the actual parameter.   
Step 4: if iskaprekar returned true then print the number to screen.   
  
**Method iskaprekar:**  
Step 1: store the square of the formal parameter in sq.   
Step 2: store the length of sq in len bu using the formula log10(sq)+1 = len.   
Step 3: store the value of sq % 10 len/2 + 1 in part1 if len is odd.   
Step 4: store the value of sq % 10 len/2 in part2 if len is even.   
Step 5: return the value of part1 + part2 == x

### Source code:

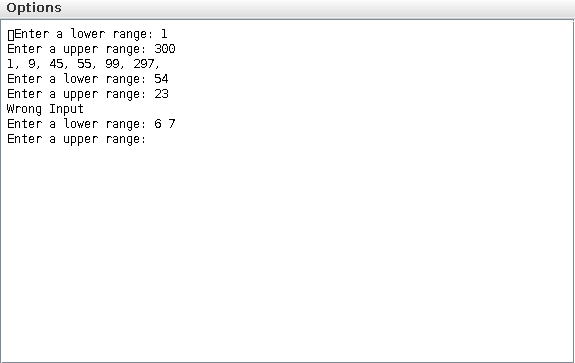
public class Kaprekar  
{  
  private int lower\_range ; // stores lower range   
  private int upper\_range ; // stores upper range   
   
  /\* initializes the whole system \*/   
  Kaprekar(int lr, int ur)   
  {   
    this.lower\_range = lr ;   
    this.upper\_range = ur ;   
  }   
   
  /\* displays the kaprekar numbers in order of their existance \*/   
  void display()   
  {   
    for(int i = lower\_range; i < upper\_range; i++)   
    {   
      if(iskapraker(i))   
        System.out.print(i+", ") ;   
    }   
    System.out.println() ;   
  }   
   
  /\* the main function that will determine if the number is really kapraker\*/   
  boolean iskapraker(int x)   
  {   
    int sq = x\*x ; // the square of the number   
    int len = (int)(Math.log10(sq)+1) ; // the total number of digits in sq   
   
    // the half of the number using mathematics   
    int part1 = sq % (int)Math.pow(10, (int)Math.floor((len \* 1.0)/2.0) + (len % 2 == 1 ? 1 : 0)) ;   
    int part2 = sq / (int)Math.pow(10, (int)Math.ceil((len \* 1.0)/2.0)) ;   
   
    //System.out.println(part2+" "+part1) ;   
   
    return (part1+part2 == x) ;   
  }   
}

   
import java.util.Scanner ;   
public class Kaprekar\_main   
{   
  /\* main entry point \*/   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in) ; // input handler   
    int lr = 0, ur = 0;   
   
    do   
    {   
      System.out.print("Enter a lower range: ") ;   
      lr = sc.nextInt() ;   
      System.out.print("Enter a upper range: ") ;   
      ur = sc.nextInt() ;   
      if(lr <= 0 || ur <= lr || ur <= 0)   
        System.out.println("Wrong Input") ;   
    }while(lr <= 0 || ur <= lr || ur <= 0) ;   
   
    Kaprekar kp = new Kaprekar(lr, ur) ;   
    kp.display() ; // main interface   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| lower\_range | it stores the lower range of number | int | object |
| upper\_range | it stores the upper range of number | int | object |
| lr | it is a temporary variable to store range of number | int | main(),Kaprekar() |
| ur | it is a temporary variable to store range of number | int | main(),Kaprekar() |
| i | iterator variable to control the for loop | int | display() |
| x | it is the input to the iskaprekar func. | int | iskaprekar() |
| sq | it stores the square of the intput | int | iskaprekar() |
| len | it is the number of digits in sq | int | iskaprekar() |
| part1 | it is the first part | int | iskaprekar() |
| part2 | it is the second part | int | iskaprekar() |
| sc | it is a input handler | Scanner | main() |
| kp | it is the control object | Kaprekar | main() |
|  |  |  |  |

### Output:



### Assignment 2

Write a Java Program to pring the first N numbers of the Pell series.  
In mathematics, the Pell numbers are an infinite sequence of integers. The sequence of Pell numbers starts with 0 and 1, and then each Pell numbers is the sum of twice the previous Pell number and the Pell number before that.:  
  
thus, 70 is the companion to 29, and 70 = 2 \* 29 + 12 = 58 + 12.  
  
The first few terms of the sequence are :  
  
0, 1, 2, 5, 12, 29, k70, 169, 408, 985, 2375, 5741, 1386

### Algorithm:

**Class Pell\_main:**  
**Method Main:**  
Step 1: create a input handler and take the number of terms as input   
Step 2: if the terms have a less number than 0 then promp user for reinput   
Step 3: create a object of Pell class and call display method  
  
**Class Pell:**  
**Method display:**  
Step 1: create two variable for first and secon pell numbers  
Step 2: start a ‘for’ loop from 0 to N with the loop control i  
Step 3: print 2 \* second term + first term to the screen and also store it in c  
Step 4: set first term to second term and second term to c.

### Source code:

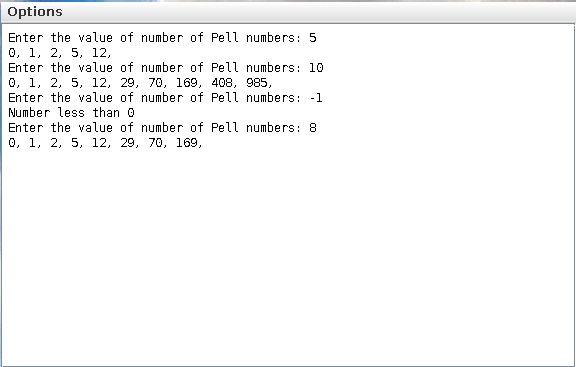
   
public class Pell   
{   
  /\* displays the whole series from start to finish \*/   
  void display(int N)   
  {   
    int a = 0 ; // starting you know right   
    int b = 1 ; // next number   
    for(int i = 0; i < N; i++)   
    {   
      System.out.print(a+", ") ;   
      // store result temporarily   
      int c = b \* 2 + a ;   
      a = b  ;   
      b = c ;   
    }   
    System.out.println() ;   
  }   
}

   
import java.util.Scanner ;   
public class Pell\_main   
{   
  /\* entery point of the whole program \*/   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in);   
    int N = 0 ;   
   
    do // I don’t want the whole program to collapse   
    {   
      System.out.print("Enter the value of number of Pell numbers: ") ;   
      N = sc.nextInt() ;   
      if(N <= 0)   
        System.out.println("Number less than 0") ;   
    }while(N <= 0) ;   
   
    Pell px = new Pell() ; // object for no damn reason at all   
    px.display(N) ; // displays series   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| sc | input handler object that is used for input | Scanner | main() |
| N | number of pell series terms | int | main(),display() |
| a | first term of pell series | int | display() |
| b | second term of pell series | int | display() |
| c | temporary variable that is used for storing tmp value of stuff | int | display() |
| px | Pell Class object used to call display() | Pell | main() |
|  |  |  |  |

### Output:



### Assignment 3

Write a program to take lower and upper range from the user and print all th eocta prime numbers within that range. (A octaprime number is a number whose octal equivalent is prim number.) Example: 15 is a octaprime number as its octal equivalent is 17 which is a prime number.

### Algorithm:

**Class OctaPrime\_main:**  
**Method Main:**  
Step 1: create a input handler and accept the lower and upper limits  
Step 2: bounds check on the input and if they fail reinput the data  
Step 3: call the printoctalprimes function for the execution of the program  
  
**Class OctaPrime\_main:**  
**Method printoctalprimes:**  
Step 1: declare i as a loop control variable   
Step 2: start a ‘for’ loop from lower\_range to upper\_range with i as the loop control.  
Step 3: get a new octal number from the octal function and i as the input  
Step 4: print the octal as a octoprime if it is a prime.  
  
**Method octal:**  
Step 1: create variable to create octal number and a stabilizer.  
Step 2: get into a while loop until x becomes zero  
Step 3: add (x % 8) \* dmz to the value of oc  
Step 4: increase dmz by 10  
Step 5: return the value of oc  
  
**Method isPrime:**  
Step 1: if x less than or equals to one then return false  
Step 2: if numbers starting from 2 till x-1 are a factor of x then return false  
Step 3: if non of the cases match return true

### Source code:

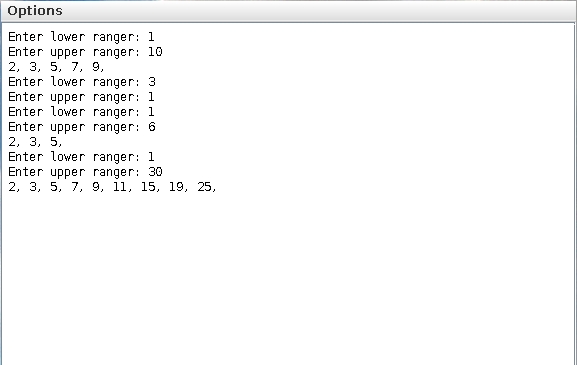
   
public class Octaprime   
{   
  /\* returns the octal variant of the number \*/   
  int octal(int x)   
  {   
    int oc = 0 ; // octal numbers   
    int dmz = 1;   
    while(x != 0)   
    {   
      oc += (x % 8) \* dmz;   
      dmz \*= 10;   
      x /= 8 ;   
    }   
    return oc ;   
  }   
   
  /\* checks if the number is prime or not \*/   
  boolean isPrime(int x)   
  {   
    if(x <= 1) return false ;   
    for(int i = 2; i < x; i++) if(x % i == 0) return false ;   
    return true ;   
  }   
   
  /\* prints the primes accorsding to the instructions \*/   
  void printoctaprimes(int lr, int ur)   
  {   
    for(int i = lr; i < ur; i++)   
    {   
      int o = octal(i) ;   
      if(isPrime(o))   
        System.out.print(i+", ") ;   
    }   
    System.out.println() ;   
  }   
}

   
import java.util.Scanner ;   
public class Octaprime\_main   
{   
  /\* entery point of the stuff \*/   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in) ;   
    int lr = 0, ur = 0 ;   
   
    do   
    {   
      System.out.print("Enter lower ranger: ") ;   
      lr = sc.nextInt() ; // lower limit   
      System.out.print("Enter upper ranger: ") ;   
      ur = sc.nextInt() ; // upper limit   
    }while(lr <= 0 || ur <= 0 || ur <= lr) ;   
   
    Octaprime op = new Octaprime() ;   
    op.printoctaprimes(lr, ur) ; // outputs the octaprimes   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| op | object to access methods | OctalPrime | main() |
| x | intput to the octal function | int | octal() |
| dmz | octal number’s position stabilizer | int | octal() |
| oc | octal number that is generated | int | octal() |
| x | intput to the is prime function | int | isPrime() |
| i | controls the for loop | int | isPrime(), printoctalprimes() |
| lr | lower range limit | int | printoctalprimes(), main() |
| ur | upper range limit | int | printoctalprimes(), main() |
| sc | input handler for input (duh) | int | main() |
| o | octal number that was generated | int | printoctalprimes() |
|  |  |  |  |

### Output:



### Assignment 4

Wriate a program in java which take as input the name of a sudent and marks of 5 subjects of the student. The avarage marks is calculated for the student. This repeated for N students. The program will display the name and the average marks of the student with the lowest average. (No sorting or searching techinque to be applied)

### Algorithm:

**Class Avarage\_main:**  
**Method Main:**  
Step 1: create a input handle to accept input from the user  
Step 2: take the input of number of students in N   
Step 3: Check if the input makes sense if not exit   
Step 4: create temporary and call displaySmall  
  
**Class Avarage:**  
**Method displaySmall:**  
Step 1: create 2 values avgl and namel to store the name and the avarage of least student  
Step 2: start a loop and continue looping until N is 0.  
Step 3: take input of 5 subjects from the stdin.  
Step 4: create a temporary variable and store the avarage of the 5 subjects.  
Step 5: check if the avarage of tmp variable is lower than the avgl.  
Step 6: if the check passes then replace the value of avgl with avg.   
Step 7: and also replace the name depending on Step 5 condition.   
Step 8: reduce N by 1.  
Step 9: after completing the whole loop print the results.

### Source code:

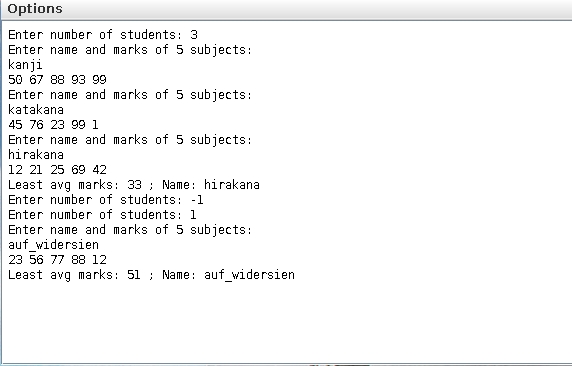
   
import java.util.Scanner ;   
public class Avarage   
{   
  /\* display smallest \*/   
  void displaySmall(Scanner sc, int N)   
  {   
    // a very big number which is way above 100 so it will accomodate   
    int avgl = 0xffffff ; /\* least avarage of the const \*/   
    String namel = "" ; /\* name of the person begin tortured by society \*/   
    while(N!=0)   
    {   
      System.out.println("Enter name and marks of 5 subjects: ") ;   
      String name = sc.next() ;   
      int a[] = {sc.nextInt(), sc.nextInt(), sc.nextInt(), sc.nextInt(), sc.nextInt()} ;   
      int avg = (a[0] + a[1] + a[2] + a[3] + a[4])/5 ;   
      if(avg < 0) return ; // best kind of err handeling   
   
      if(avg < avgl)   
      {   
        avgl = avg ;   
        namel = name ;   
      }   
   
      N-- ;   
    }   
   
    System.out.println("Least avg marks: "+avgl+" ; Name: "+namel) ;   
  }   
}

   
import java.util.Scanner ;   
public class Avarage\_main   
{   
  /\* entery point like who in the right mind would have guessed\*/   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in) ;   
    System.out.print("Enter number of students: ") ;   
    int N = sc.nextInt() ;   
    if(N < 0) return ; // best kind of err handeling   
    new Avarage().displaySmall(sc, N) ;   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| avgl | least average of the cost | int | displaySmall |
| namel | name of the person with smallest number | String | displaySmall |
| a | stores the value of all 5 subject marks | int[] | displaySmall |
| avg | temporary avarage of the 5 subjects | int | displaySmall |
| N | number of students input | int | displaySmall, main |
| sc | input handler using scanner | Scanner | displaySmall, main |
|  |  |  |  |

### Output:



### Assignment 5

Vrite a program which takes N integers from the useer in an array and removes the duplicate elements from the array and display the new array.  
Example:  
Enter no of integgers : 7  
Input array elements: 1,2,3,1,2,3,4

Output

### Algorithm:

**Class Duplicate\_main:**  
**Method Main:**  
Step 1: create a input handle   
Step 2: take the number of elements as input   
Step 3: fill up the array with input data   
Step 4: create a temporary object and call the apropriate method   
  
**Class Duplicate:**  
**Method removeDupAndPrint:**  
Step 1: create a string to store out put   
Step 2: Iterate through the array.   
Step 3: Check the first index of the element in out  
Step 4: if it exists in out then don’t add else add it.  
Step 5: replace the “[space]” with “,[space]” and print it

### Source code:

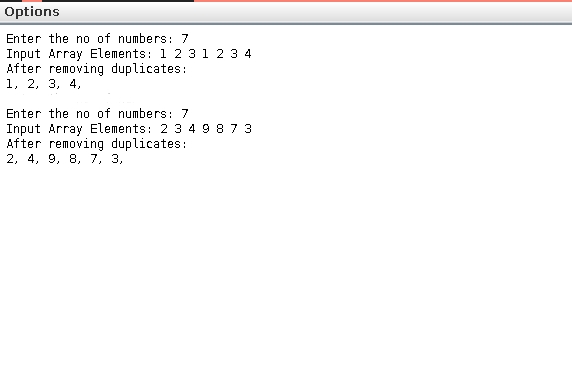
   
public class Duplicate   
{   
  void removeDupAndPrint(int x[])   
  {   
    String out = "" ; // output   
   
    // finds out the elements that are duplicate   
    for(int i = 0; i < x.length; i++)   
    {   
      // index of returs -1 if element does not exist   
      if(out.indexOf(x[i]+"") < 0)   
        out = x[i]+" "+out ;   
    }   
   
    // prints with a ,   
    System.out.println(out.replace(" ", ", ")) ;   
  }   
}

   
import java.util.Scanner ;   
public class Duplicate\_main   
{   
  // entry point of Duplicate\_main   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in) ; // input handler   
    int num = sc.nextInt() ; // number of elements of array because java is retarded   
    int nx[] = new int[num] ; // the array because java does not do it for me   
    while(num-- != 0) nx[num] = sc.nextInt() ; // intputs the numbers required   
   
    // did all the array resizing and printing so nothing left to do   
    new Duplicate().removeDupAndPrint(nx) ;   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| sc | inputt handler | Scanner | main |
| num | number of elements of array | int | main |
| nx | the array of input data | int[] | main |
|  |  |  |  |

### Output:



### Assignment 6

A class Mixer has been defined to merge two sorted integer array in ascending order. Some of the members of the class are given below:

|  |  |
| --- | --- |
| **Class Name:** | **Mixer** |
| **Data members/instance variables:** |  |
| int arr[] | : to store elements of an array |
| int n | : to store the size of the array |
| **Member functions:** |  |
| Mixer(int nn) | : constructor to assign n = nn |
| void accept() | : to accept the elements of the array in |
|  | ascending order without any duplicates |
| Mixer mix(Mixer A) | : to merge the current object array |
|  | elements with the parameterized array elements |
|  | and return the rusultant object. |
| void display() | : to display elements of the array |

### Algorithm:

**Class Mixer\_main:**  
**Method Main:**  
Step 1: create a input handler using Scanner calss.  
Step 2: create 2 variables to hold the length of the mixer arrays  
Step 3: check if the input is sensible and greatere than 0  
Step 4: if the check fails propmt the user again to input number  
Step 5: create two objects of mixer class using the previously declared variables.  
Step 6: accept input from the user using the accept() method.  
Step 7: call the mix method using the mix objects that are created.  
Stpe 8: display the mixed arrays.  
  
**Class Mixer:**  
**Method Mixer:**  
Step 1: set the object variable n to nn   
Step 2: create a array of size n and store it in object variable to arr  
  
**Method mix:**  
Step 1: create a new object for mixer with the size of the two input mixer arrays.  
Step 2: fill the newly created array with the elements of A and this.  
Step 3: sort the array of the newly created object using insertion sort.  
Step 4: return the object.  
  
**Method accept:**  
Step 1: create a input handler using Scanner class.  
Step 2: loop throught the arr array of the current object and fill it with input.  
  
**Method display:**  
Step 1: loop througth the whole arr array of current object and print the elements.  
Step 2: print a newline for asthetics.

### Source code:

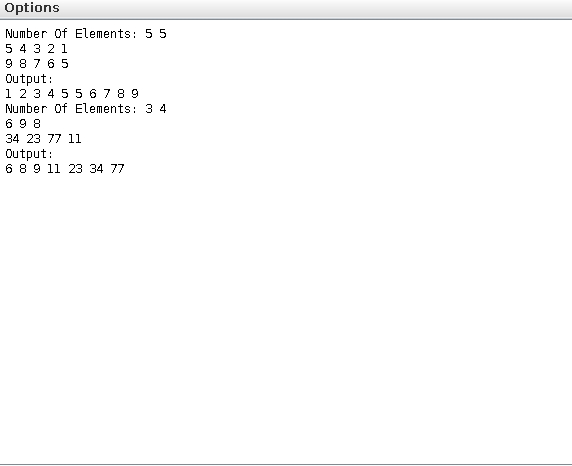
   
import java.util.Scanner ;   
public class Mixer   
{   
  int arr[] ; // to store elements of an array   
  int n ; // to store the size of the array   
   
  Mixer(int nn)   
  {   
    this.n = nn ;   
    this.arr = new int[nn] ;   
  }   
   
  // accepts all the input in the arr array   
  void accept()   
  {   
    Scanner sc = new Scanner(System.in) ;   
    for(int i = 0; i < this.n; i++)   
    {   
      this.arr[i] = sc.nextInt() ; // int is the input type   
    }   
  }   
   
  // mixes stuff like merge sort.   
  Mixer mix(Mixer A)   
  {   
    Mixer m = new Mixer(this.n + A.n) ;   
   
    // copy every thing to the m.arr first   
    for(int i = 0; i < this.n; i++) m.arr[i] = this.arr[i] ;   
    for(int i = this.n; i < m.n; i++) m.arr[i] = A.arr[i-this.n] ;   
   
    // now sort the whole thing.   
    for(int i = 1; i < m.n; ++i)   
    {   
      int key = m.arr[i] ;   
      int j = i - 1 ;   
      while(j >= 0 && m.arr[j] > key)   
      {   
        m.arr[j+1] = m.arr[j] ;   
        --j ;   
      }   
      m.arr[j+1] = key ;   
    }   
    return m ;   
  }   
   
  // print the whole beautiful \*sigh\* array   
  // I had to write the same thing 10 times in a row   
  // now I am tired. god save my soul.   
  void display()   
  {   
    for(int i = 0; i < this.n; i++)   
      System.out.print(this.arr[i]+" ") ;   
    System.out.println() ;   
  }   
}

   
import java.util.Scanner ;   
public class Mixer\_main   
{   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in) ;   
    int n1 = 0; // two different elements   
    int n2 = 0;   
   
    System.out.print("Number Of Elements: ") ;   
   
    do{ n1 = sc.nextInt() ;}while(n1 <= 0) ; // inputs of both   
    do{ n2 = sc.nextInt() ;}while(n2 <= 0) ;   
   
    Mixer m1 = new Mixer(n1) ; // initializing 2 objs   
    Mixer m2 = new Mixer(n2) ;   
   
    m1.accept() ; // accepts input in mixer   
    m2.accept() ;   
   
    Mixer m3 = m1.mix(m2) ; // mixes the stuff   
   
    System.out.println("Output: ") ; // prints the stuff out   
    m3.display() ;   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| n1 | number of elements | int | main |
| n2 | number of elements | int | main |
| m1 | object to be filled | Mixer | main |
| m2 | object to be filled | Mixer | main |
| m3 | output of the mixers | Mixer | main |
| arr | to store elements of an array | int[] | Mixer |
| n | to store size of arr | int | Mixer |
| nn | temporary input var | int | Mixer() |
| sc | input of all the numbers | Scanner | accept, main |
| A | mix object to be used to mix stuff | Mixer | mix |
| m | output of the mix method | Mixer | mix |
| i,j | loop control variables | int | mix |
| key | key of insertion sort | int | mix |
|  |  |  |  |

### Output:



### Assignment 7

Write a program which takes n integers as input(max 50 integers) and stores them in an array data from index 0 to n-1. Now we want to rearrange the integers in the following way:- find the minmum value and put it in position (n/2) [for odd number of elements] oand position (n/2-1) [for even number number elements]) ; find the second smallest value and put it to its right; then the third small and place it to its lieft and so on altering right and left until all the integers are done.  
  
Initial array:- (size: 7)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 7 | 3 | 1 | 6 | 4 | 2 |
|  |  |  |  |  |  |

Intial array:- (size: 6)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 7 | 3 | 1 | 6 | 4 | 2 | 5 |
|  |  |  |  |  |  |  |

After re-arrangement of the first array becomes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 6 | 3 | 1 | 2 | 4 | 7 |
|  |  |  |  |  |  |

After re-arrangement the second array becomes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| 7 | 5 | 3 | 1 | 2 | 4 | 6 |
|  |  |  |  |  |  |  |

### Algorithm:

**Class Alternate\_main:**  
**Method Main:**  
Step 1: create a input handle from the console  
Step 2: create variables to hold array and sizes  
Step 3: take input from user if the input is bad repromt  
Step 4: call fill method of the Alternate class.  
Step 5: print the nx buff.  
  
**Class Alternate:**  
**Method fill:**  
Step 1: create a variable called pos. Step 2: check if intput is array length is even Step 3: if the check passes then set pos to arr.length/2 - 1 Step 3: else set pos to arr.length/2 Step 4: loop from 0 to arr.length using i Step 5: loop from 0 to arr.length - 1 using j Step 6: swap arr[j] and arr[j+1] if arr[j] ¿ arr[j+1] Step 7: after loop start new loop from 0 to nx.length using i Step 8: set pos = pos - i \* (1 if i is even else -1) Step 9: set nx[pos] = arr[i] Step 10: complete loop.

### Source code:

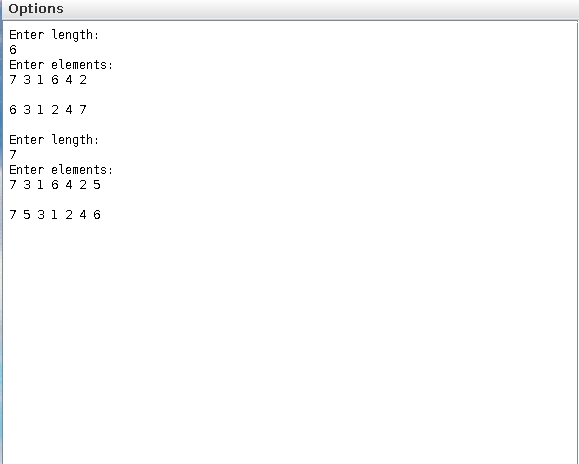
   
public class Alternate   
{   
  // fills the whole array   
  void fill(int arr[], int nx[])   
  {   
    int pos = arr.length % 2 == 0 ? arr.length/2 - 1: arr.length/2 ;   
    int dn = 1 ;   
   
    // sorting the whole thing using insertion sort.   
    for(int i = 0; i < arr.length; i++)   
    {   
      for(int j = 0; j < arr.length - 1; j++)   
      {   
        if(arr[j] > arr[j+1])   
        {   
          int tmp = arr[j] ;   
          arr[j] = arr[j+1] ;   
          arr[j+1] = tmp ;   
        }   
      }   
    }   
   
    // goes up 2 times because other wise things might result   
    // in SIGSEGV which is what i dont’t like.   
    for(int i = 0; i < nx.length; i++)   
    {   
      pos -= i \* (int)Math.pow(-1, i) ;   
      nx[pos] = arr[i] ;   
    }   
  }   
}

   
import java.util.Scanner ;   
public class Alternate\_main   
{   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in) ;   
    int n = 0 ;   
    int arr[] = null ;   
    int nx[] = null ;   
   
    do{ n = sc.nextInt() ;}while(n <= 0) ;   
    arr = new int[n] ;   
    nx = new int[n] ;   
   
   
    for(int i = 0; i < n; i++)   
      arr[i] = sc.nextInt() ;   
   
    // arr will be sorted after this but who cares   
    new Alternate().fill(arr, nx) ;   
   
    // printing i guess   
    System.out.println() ;   
    for(int i = 0; i < n; i++)   
      System.out.print(nx[i]+" ") ;   
    System.out.println() ;   
  }   
   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| arr | the input array which gets sorted | int[] | fill,main |
| nx | the buffer that is to be filled | int[] | fill,main |
| pos | the starting position of the filler | int | fill |
| i, j | loop control variable of the sorter | int | fill, main |
| tmp | temporary variable for swapping | int | fill |
| n | number of elements in the array | int | main |
|  |  |  |  |

### Output:



### Assignment 8

Two matrices are said to be equal if they have the same dimention and their corresponding elements are equal.  
For examples, the two matrix A and B given below are equal:

|  |  |
| --- | --- |
| Matrix A | Matrix B |
| |  |  |  | | --- | --- | --- | |  |  |  | | 1 | 2 | 3 | |  |  |  | | 2 | 4 | 5 | |  |  |  | | 3 | 5 | 6 | |  |  |  | | |  |  |  | | --- | --- | --- | |  |  |  | | 1 | 2 | 3 | |  |  |  | | 2 | 4 | 5 | |  |  |  | | 3 | 5 | 6 | |  |  |  | |

Design of class EqMat to check if two matrices are equal or not. Assume that the two matrices have the same dimention. some of the membersof the class are given below:

|  |  |
| --- | --- |
| **Class name** | **EqMat** |
| **Data members/instance** |  |
| **Members:** |  |
| a[][] | to store integer elements |
| m | to store number of rows |
| n | to store number of columns |
| **Members functions methods:** |  |
| EqMat(int mm, int nn) | parameterized constructor initialize the data |
|  | members m = mm and n = nn ; |
| void readarray() | to enter elemtents in the array |
| int check(EqMat p, EqMat q) | checks if the parametarized objects p and q |
|  | are equal and returns 1 if true otherwise returns 0. |
| void print() | displays the arrays elemtents. |

Define the class EqMat giving details of constructor(), void readarray(), int check(EqMat, EqMat) and void print(). Define the main() function to create objects and call the funcitons accordingly to enable the task.

### Algorithm:

**Class EqMat\_main:**  
**Method main:**  
Step 1: take input from the user about the row and cols.  
Step 2: check if the input is greater than 0 and retake input if not.  
Step 3: create a EqMat object and call readarray for input array.  
Step 4: repeat the last three steps again for a new object.  
Step 5: call print function of the first object.  
Step 6: call print function of the second object.  
Step 7: check if a.check(b) is equal to 1  
Step 8: if the check passes print array are equal  
Step 9: if the check fails print array are not equal  
  
**Class EqMat:**  
**Method Eqmat:**  
Step 1: initialize all object variables  
Step 2: create a new object for object a variable  
  
**Method check:**  
Step 1: check if the matrix are even same dimentional.  
Step 2: if the check fails return 0 ;   
Step 3: loop from 0 to p.m using i as variable  
Step 4: loop from 0 to q.n using j as variable   
Step 5: check if p.a[i][j] is not equal to q.a[i][j].   
Step 6: if the check fails return 0 ;  
Step 7: return if the control makes it out of the loops.  
  
**Method print:**  
Step 1: loop from 0 through the number of rows using i as var.  
Step 2: loop from 0 through the number of cols using j as var.  
Step 3: print a[i][j] with a space at the end.  
Step 4: print a new line at the end of j var.  
  
**Method readarray:**  
Step 1: create a input stream handle.  
Step 2: take input in the buffer created in a.

### Source code:

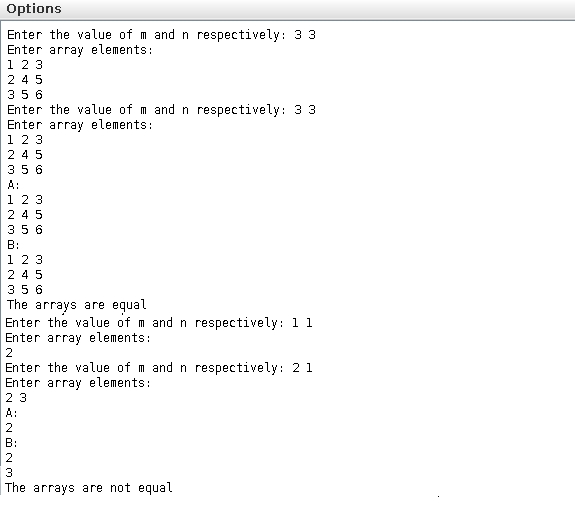
   
import java.util.Scanner ;   
public class EqMat   
{   
  int a[][] ; // integer matrix storer   
  int m ; // number of rows   
  int n ; // number of cols   
   
  // init function of java called the costructor   
  EqMat(int mm, int nn)   
  {   
    this.m = mm ;   
    this.n = nn ;   
    this.a = new int[mm][nn] ;   
  }   
   
  // check if the whole thing works. it returns 1 if ok else 0   
  int check(EqMat p, EqMat q)   
  {   
    if(p.m != q.m || p.n != q.n) return 0 ;   
    for(int i = 0; i < p.m; i++)   
    {   
      for(int j = 0; j < q.n; j++)   
      {   
        // if one is not equal and thus it does not waste time   
        if(p.a[i][j] != q.a[i][j])   
          return 0 ;   
      }   
    }   
    return 1 ;   
  }   
   
  // print the whole thing   
  void print()   
  {   
    // loops through all the elements and prints the whole thing.   
    for(int i = 0; i < this.m; i++)   
    {   
      for(int j = 0; j < this.n; j++)   
      {   
        System.out.print(this.a[i][j]+" ") ;   
      }   
      System.out.println() ;   
    }   
  }   
   
  // reads matrix input from the start.   
  void readarray()   
  {   
    Scanner sc = new Scanner(System.in) ;   
    for(int i = 0; i < m; i++)   
    {   
      for(int j = 0; j < n; j++)   
      {   
        a[i][j] = sc.nextInt() ;   
      }   
    }   
  }   
}

   
import java.util.Scanner ;   
public class EqMat\_main   
{   
  // entry point   
  public static void main(String args[])   
  {   
    int m = 0;   
    int n = 0;   
    Scanner sc = new Scanner(System.in);   
   
    // first input   
    do   
    {   
      System.out.print("Enter the value of m and n respectively: ") ;   
      m = sc.nextInt() ;   
      n = sc.nextInt() ;   
    }while(m <= 0 || n <= 0) ;   
   
    EqMat a = new EqMat(m, n) ;   
    System.out.println("Enter array elements: ") ;   
    a.readarray() ;   
   
    // reusing the variable because I have no shame   
    do   
    {   
      System.out.print("Enter the value of m and n respectively: ") ;   
      m = sc.nextInt() ;   
      n = sc.nextInt() ;   
    }while(m <= 0 || n <= 0) ;   
   
    EqMat b = new EqMat(m, n) ;   
    System.out.println("Enter array elements: ") ;   
    b.readarray() ;   
   
    System.out.println("A: ") ;   
    a.print() ;   
   
    System.out.println("B: ") ;   
    b.print() ;   
   
    // prints relevant message if required.   
    if(a.check(a, b) == 1)   
      System.out.println("The arrays are equal") ;   
    else   
      System.out.println("The arrays are not equal") ;   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| a[][] | integer matrix storer | int[][] | EqMat:object |
| m | number of rows | int | EqMat:object, main |
| n | number of cols | int | EqMat:object, main |
| mm | a temporary variable for m | int | EqMat |
| nn | a temporary variable for n | int | EqMat |
| p | input for check function | EqMat | check |
| q | input for check function | EqMat | check |
| i | loop control varible | int | print |
| j | loop control varible | int | print |
| sc | Input handle varable | Scanner | readarray, main |
| a,b | object for testing | main | main |
|  |  |  |  |

### Output:



### Assignment 9

Write a program to declare a square matrix A[][] of order (M x M) where ’M’ is the number of rows and the number of colums such that M must be greater than 2 and less than 20. Allow the user to input integers into this matrix. Display appropriate error message for an invalid input. Perform the following tasks:  
  
a) Display the input matrix.  
  
b) Create a mirror image matrix.  
  
c) Display the mirror image matrix  
  
Test your program with the sample data and some random data:  
  
**Example 1:**  
  
INPUT : M = 3

|  |  |  |
| --- | --- | --- |
| 4 | 16 | 12 |
| 8 | 2 | 14 |
| 4 | 1 | 3 |

OUPUT :

ORIGINAL MATRIX

|  |  |  |
| --- | --- | --- |
| 4 | 16 | 12 |
| 8 | 2 | 14 |
| 4 | 1 | 3 |

MIRROR IMAGE MATRIX

|  |  |  |
| --- | --- | --- |
| 12 | 16 | 4 |
| 14 | 2 | 8 |
| 3 | 1 | 4 |

### Algorithm:

**Class Mirror\_main:**  
**Method Main:**  
Step 1: create a input handle using scanner  
Step 2: input the number of chracters  
Step 3: create a 2d-array and fill it with data.  
Step 4: check and repromt if the size if greater than 22 or lesser than 2.  
Step 5: create matrix object.  
Step 6: display the matrices.  
  
**Class Mirror:**  
**Method display\_original:**  
Step 1: loop from 0 to M using i  
Step 2: loop from 0 to M using j  
Step 3: print A[i][j]  
Step 4: after second loop print new line  
  
**Method display\_mirror:**  
Step 1: loop from 0 to M using i  
Step 2: loop from 0 to M using j  
Step 3: print A[i][M-1-j]   
Step 4: after second loop print new line  
  
**Method Mirror:**  
Step 1: take M and A[][] as a input.  
Step 2: initialize object a new buffer called A with the size M x M.  
Step 3: fill the index of object’s A[i/M][i%M] = A[i/M][i%M]  
Step 4: initialize object’s M to be M

### Source code:

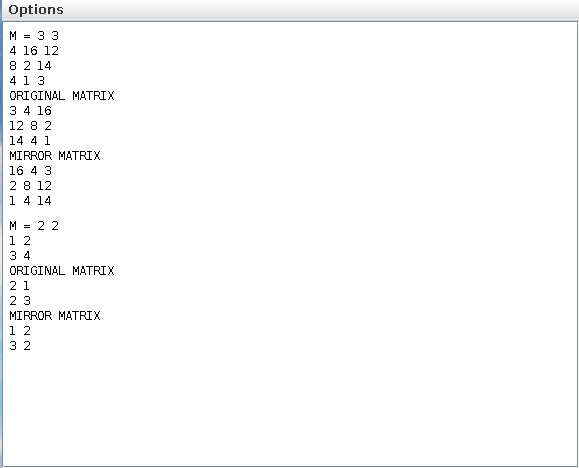
   
public class Mirror   
{   
  int A[][] ; // the matrix storage.   
  int M ; // the size of the matrix.   
   
  // Did i really have to give a comment here   
  // is the function name not at all descriptive   
  // :( Sorry but it hurts me.   
   
  Mirror(int M, int A[][])   
  {   
    this.A = new int[M][M] ;   
   
    // this deep copies the array and   
    // does not store the array reference   
    // and thus it does not change the   
    // value of the input array.   
    for(int i = 0; i < M\*M; i++)   
      this.A[i/M][i%M] = A[i/M][i%M] ;   
   
    this.M = M ;   
  }   
   
  // prints the original matrix   
  void display\_original()   
  {   
    for(int i = 0; i < M; i++)   
    {   
      for(int j = 0; j < M; j++)   
      {   
        System.out.print(A[i][j]+" ") ;   
      }   
      System.out.println() ;   
    }   
  }   
   
  // mirrors the matrix prints the elements   
  void display\_mirror()   
  {   
    for(int i = 0; i < M; i++)   
    {   
      for(int j = 0; j < M; j++)   
      {   
        // reverses the order of the output counting   
        // thus mirroring the matrix effectively   
        System.out.print(A[i][M-1-j]+" ") ;   
      }   
      System.out.println() ;   
    }   
  }   
}

   
import java.util.Scanner ;   
public class Mirror\_main   
{   
  // main entry point.   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in) ;   
    System.out.println("M= ") ;   
    int M = sc.nextInt() ;   
    int A[][] = new int[M][M] ;   
   
    // as the intput is asked :D   
    if(M < 2 || M > 22)   
    {   
      System.out.println("SIZE OUT OF RANGE");   
    }   
   
    // intput of the array as well   
    for(int i = 0; i < M\*M; i++)   
      A[i/M][i%M] = sc.nextInt() ;   
   
    Mirror mr = new Mirror(M, A) ;   
    System.out.println("ORIGINAL MATRIX") ;   
    mr.display\_original() ;   
    System.out.println("MIRROR MATRIX") ;   
    mr.display\_mirror() ;   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| A | the matrix storage | int[][] | Mirror:object, Mirror, main |
| M | the size of the matrix | int[][] | Mirror:object, Mirror, main |
| i,j | index value of the for loops | display\_original, display\_mirror |  |
| sc | input handler | Scanner | main |
| mr | input Mirror object | Mirror | main |
|  |  |  |  |

### Output:



### Assignment 10

A company manufactures packingg cartons in four sizes, i.e. cartons to accommodate 6 boxes, 12 boxes, 24 boxes and 48 boxes. Desiggn a program to accept the number of boxes to be packed (N) by the user (maximum up to 1000 boxes) and display the break-up of the cartons used in descending order of capacity (i.e. preference should be given to the highest capacity available, adn if boxes left are less than 6, an extra carton of capacity 6 should be used.)  
  
Test your program with the following data and some random data:  
  
**Example 1:**  
  
**INPUT:** N = 726  
**OUTPUT:** 48 x 15 = 720  
6 x 1 = 6  
Remaining boxes = 0  
Total number of boxes = 726  
Toatl number of cartons = 16  
  
**Example 2:**  
  
**INPUT:** N = 726  
**OUTPUT:** 48 x 15 = 720  
6 x 1 = 6  
Remaining boxes = 0  
Total number of boxes = 726  
Toatl number of cartons = 16

### Algorithm:

**Class Company\_main:**  
**Method Main:**  
Step 1: create a input handle using Scanner class  
Step 2: take input the number of boxes.  
Step 3: check if the input is sane. if the input is not sane print INVALID INPUT.  
Step 4: create a compmany object using the N as intput.  
Step 5: call calculate using company object.  
Step 6: call pretty\_print using company object.  
  
**Class Company:**  
**Method calculate:**  
Step 1: create a copy of N locally.  
Step 2: loop from 0 to this.cartons.length using the variable i  
Step 3: execute the statement num\_cartons[i] = N / cartons[i]  
Step 4: execute the statement num\_carton++  
Step 5: execute the statement N %= cartons[i]  
Step 6: set remainder to N  
  
**Method pretty\_print:**  
Step 1: loop from 0 to this.num\_cartons.lengths using the variable i  
Step 2: execute print statement only if num\_cartons[i] not equals 0  
Step 3: print the output in a fancy format.  
  
**Method Company:**  
Step 1: initialize cartons with [48 24 12 6] for carton listings.  
Step 2: initialize num\_cartons with a new buffer of number of type of cartons  
Step 4: initialize N with input N.  
Step 5: initialize num\_carton to zero

### Source code:

   
public class Company   
{ 

int N ; // the number of boxes   
 int cartons[] ; // the varid capacity boxes list   
 int num\_cartons[] ; // the magnitude of each boxes   
 int num\_carton ; // this is the total number of cartons that are required   
 int remainder ; // the remaining boxes after filling   
   
  // creates the boxes and stuff   
  Company(int N)   
  {   
    this.cartons = new int[]{48, 24, 12, 6} ;   
    this.num\_cartons = new int[this.cartons.length] ;   
    this.N = N ;   
    this.num\_carton = 0 ;   
  }   
   
  void calculate()   
  {   
    // saves a local copy to calculate data   
    int N = this.N ;   
   
    // rolls through all the cartons and calculates their magnitude.   
    for(int i = 0; i < this.cartons.length; i++)   
    {   
      // gets how many can be acomodated in number of cartons   
      this.num\_cartons[i] = N / this.cartons[i] ;   
      this.num\_carton++ ;   
      // then cheks how many remained.   
      N %= this.cartons[i] ;   
    }   
   
    // checks if there are any remainders.   
    this.remainder = N ;   
  }   
   
  // this function prints the whole thing in a fancy manner. i like it :)   
  void pretty\_print()   
  {   
    // the print statements are tweaked many times to get the correct result.   
    for(int i = 0; i < this.num\_cartons.length; i++)   
    {   
      if(this.num\_cartons[i] != 0)   
        System.out.println("\t\t\t"+this.cartons[i]+" x "+this.num\_cartons[i]+"\t= "+   
          (this.cartons[i]\*this.num\_cartons[i])) ;   
    }   
    System.out.println("Remaining boxes \t"+((this.remainder != 0)?this.remainder+" x 1\t= ":"\t= ")+this.remainder) ;   
    System.out.println("Total Number of boxes \t\t= "+this.N) ;   
    System.out.println("Total Number of cartons \t= "+this.N) ;   
  }   
}

   
import java.util.Scanner ;   
public class Company\_main   
{   
  // entry point of the program   
  public static void main(String args[])   
  {   
    Scanner sc = new Scanner(System.in) ;   
    System.out.print("N = ") ;   
    int N = sc.nextInt() ;   
    if(N < 0 || N > 1000)   
    {   
      System.out.println("INVALID INPUT") ;   
      return ;   
    }   
    Company c = new Company(N) ;   
    c.calculate() ;   
    c.pretty\_print() ;   
  }   
}

### Variable Listing:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Name** | **Function** | **Type** | **Scope** |
|  |  |  |  |
| sc | input handler object that is used for input | Scanner | main |
| N | number of box | int | main,Company:obj,calculate |
| c | Company obbject createor | Company | main |
| i,j | iterator control variable | pretty\_print | pretty\_print, calculate |
| cartons | the varid capacity boxes list | int[] | Company:obj |
| num\_cartons | the magnitude of each boxes | int[] | Company:obj |
| num\_carton | this is the total number of cartons that are required | int | Company:obj |
| remainder | the remaining boxes after filling | int | Company:obj |
|  |  |  |  |

### Output:

