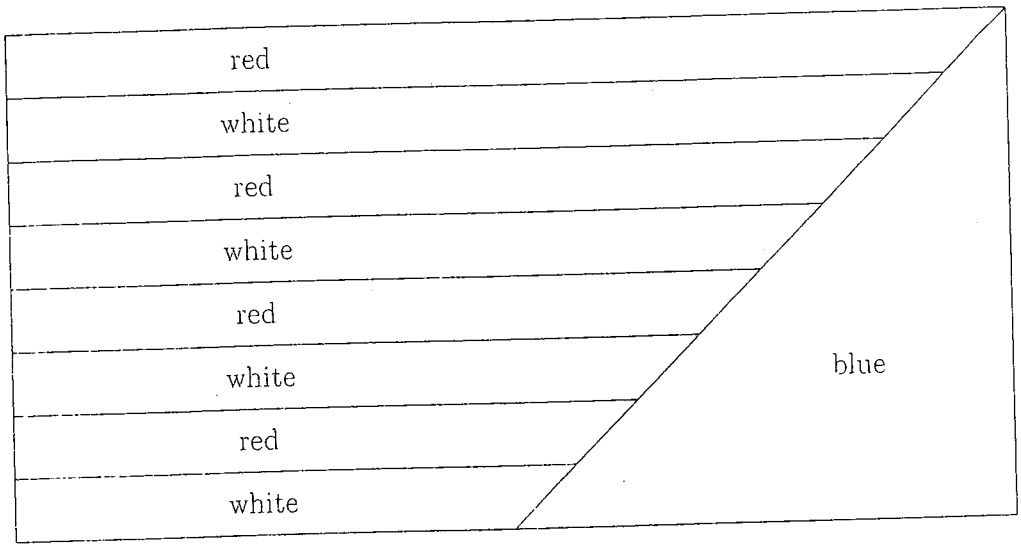
1.

The Noland national flag is a rectangle with a width:height ratio of 2:1 and showing the following pattern.



Write a method

```
public void drawNoland(int n)
```

that draws the Noland flag on the screen with a height of n pixels. Your method should create and use a `SimpleCanvas` (as used in lectures and laboratories) to draw on. All of the colours needed are pre-constructed `Color` objects.

PUT YOUR ANSWER ON THE NEXT PAGE

2.

(a) The speed limit on the freeway to Noland is 100km/h. The following table summarises the penalties which apply for exceeding this limit under various circumstances.

Speed (s)	First offence?	Penalty
$s \leq 100$	not applicable	\$0
$100 < s < 120$	yes	\$105
$120 \leq s$	yes	\$160
$100 < s$	no	\$200

Write a method

```
public String speedResponse(int s, boolean f)
```

that returns the message and penalty that apply in a given situation.

(5 marks)

(b) Write a method

```
public boolean alternating(boolean[] a)
```

that returns `true` if and only if a contains no adjacent elements which have the same value.

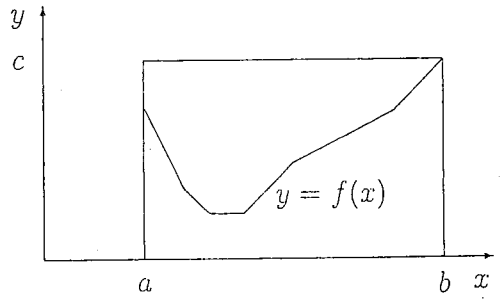
For example `alternating({false, true, false})` returns `true`,
but `alternating({true, false, false})` returns `false`.

(5 marks)

3.

Given a function `f(x) = y` such that `0 <= y <= c` whenever `a <= x <= b`, we wish to estimate the area under `f` using a sampling procedure.

Imagine the curve for `f` is enclosed in a rectangle with width `b - a` and height `c`.



Generate `n` points randomly in this rectangle, and count how many points lie under the curve. The proportion of points that lie under the curve approximates `A/B`, where `A` is the area under the curve, and `B` is the area of the rectangle. From this we can estimate `A`. Clearly the precision of the estimate increases with the number of samples.

Given a method with the signature

```
public double f(double x)
```

write a method

```
public double area(double a, double b, double c, int n)
```

that implements the above procedure for `f`.

You should use an object from the library class `java.util.Random` to generate a sequence of random points inside the rectangle, and you will need the following method from that class.

```
public double nextDouble()

    Returns the next pseudorandom, uniformly-distributed double
    between 0.0 and 1.0 from this random number generator's sequence.
```

PUT YOUR ANSWER ON THE NEXT PAGE

4.

Consider the class BankAccount, for use in a bank's account record system.

```
public class BankAccount {

  private String accName; // the account holder's name, e.g. Bill Gates
  private int accNumber;
  private int balance;

  public String getName()
  {return accName;}

  public int getBalance()
  {return balance;}

  // constructor and other details omitted
}
```

Write an efficient method

```
public int range(BankAccount[] accList)
```

that returns the difference between the largest and smallest balances in `accList`. `range` should throw an exception if `accList` is empty or `null`.

You may assume that the records are sorted by their account numbers.

PUT YOUR ANSWER ON THE NEXT PAGE

5.

A *word search* puzzle gives the player a rectangular grid of letters g and a word `w` to find in the grid.

q	*s*	*w*	*w*	*w*	*o*	*w*	*a*	*l*
h	*e*	*w*	*a*	*w*	*a*	*w*	*a*	*g*
e	*r*	*l*	*y*	*l*	*a*	*g*	*r*	*w*
q	*e*	*w*	*a*	*l*	*e*	*s*	*c*	*d*
s	*w*	*a*	*k*	*v*	*b*	*s*	*s*	*e*
a	*a*	*l*	*r*	*h*	*b*	*n*	*m*	*x*
l	*l*	*e*	*e*	*h*	*k*	*s*	*d*	*e*
e	*e*	*i*	*l*	*o*	*v*	*e*	*w*	*c*
s	*s*	*l*	*y*	*n*	*d*	*o*	*n*	*a*

The goal is to find out whether `w` occurs anywhere in g. For our purposes we will recognise `w` if it occurs in g either

- left-to-right, or
- downwards, or
- left-to-right and downwards, i.e. diagonally down and across g.

Thus the word *wales* occurs exactly three times in the above grid. Write a method

```
public boolean wordsearch(String w, char[][] g)
```

that returns `true` if and only if the word `w` is found in the grid g according to the above rules.

You may assume that g is rectangular. You will benefit from writing one or more private helper methods to decompose the problem.

PUT YOUR ANSWER ON THE NEXT PAGE

6.

(a) Write a method

```
public boolean subset(int[] a, int[] b)
```

that returns `true` if and only if every element of `a` also occurs in `b`.

For example `subset({1,2,1},{4,2,3,1})` returns `true`,
but `subset({1,5,1},{4,2,3,1})` returns `false`.

(5 marks)

(b) Write a method

```
public int[][] pairs(int[] a)
```

that returns a 2D array containing all pairs of adjacent elements from a.
For example `pairs({1,2,3,4})` returns `{{1,2},{2,3},{3,4}}`.

(5 marks)