# Programming Tasks

All tasks are performed without Internet Access and support material. Only the Development tool can be used.
You are free to use any kind of programming language and Development tool.

Total test time 150 minutes

**Task 1**

How do you swap two integers without using temp variable?
Assume you have two integers I = 20 and J = 34.
Your task is to write a program which demonstrates how you swap the variables without using any additional variable so that J = 20 and I = 34.
A solution which handles integer overflow gives an extra bonus.

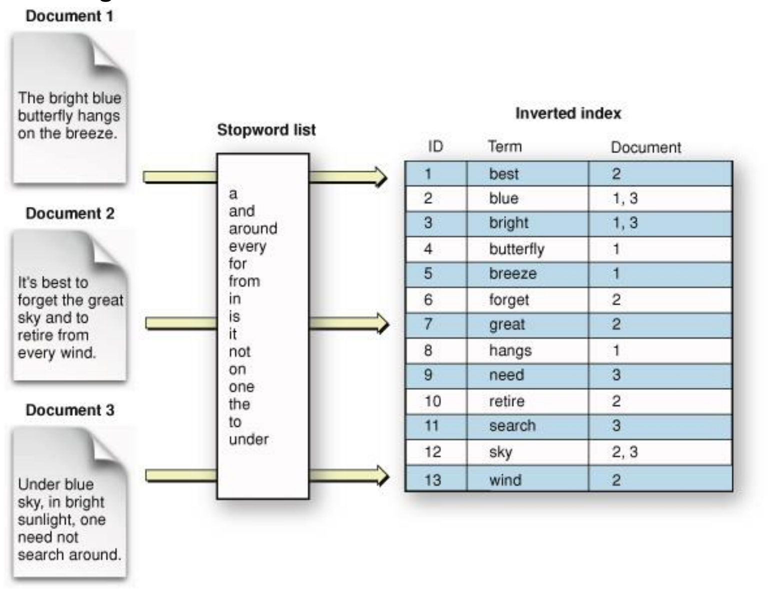
**Task 2**
Write a program that outputs all possibilities to put + or - or nothing between the numbers 1, 2, ..., 9 (in this order) such that the result is always 100. For example: 1 + 2 + 34 – 5 + 67 – 8 + 9 = 100.

**Task 3**
Write Algorithms to Check if Two String are Anagram
An anagram is something where length and character matches but not the order like Army and Mary, both have the same number of characters.

**Task 4**
Write a program to find repeated characters in a String. For example, if given input to your program is "Java", it should print all duplicates characters, i.e. characters appear more than once in String and their count e.g. a = 2 because character 'a' has appeared twice in String "Java".¨
Example of output:
List of duplicate characters in String 'Programming'
g : 2
r : 2
m : 2
List of duplicate characters in String 'Combination'
n : 2
o : 2
i : 2

**Task 5**
Implement and search in an Inverted Index. You are free to use any kind of data to demonstrate the solution.
What is Inverted List?
The objective of an inverted index is to permit quick full-text searches. What’s more, they can do so at a cost of increasing processing whenever a document goes on the database. The inverted file may very

well be the database file itself instead of its index. It is, inarguably, the most popular data structure that document retrieval systems use. It is especially useful on a large scale; in search engines, for instance. The image below demonstrates the basics.


Document 1
The bright blue butterfly hangs on the breeze.
Document 2
It's best to forget the great sky and to retire from every wind.
Document 3
Under blue sky, in bright sunlight, one need not search around.
Stopword list
a
and
around
every
for
from
in
is
it
not
on
one
the
to
under
Inverted index

| ID | Term | Document |
|---|---|---|
| 1 | best | 2 |
| 2 | blue | 1, 3 |
| 3 | bright | 1, 3 |
| 4 | butterfly | 1 |
| 5 | breeze | 1 |
| 6 | forget | 2 |
| 7 | great | 2 |
| 8 | hangs | 1 |
| 9 | need | 3 |
| 10 | retire | 2 |
| 11 | search | 3 |
| 12 | sky | 2, 3 |
| 13 | wind | 2 |

Create a set of text files with words, implement a program to create an inverted index.
Also create a simple web user interface to do a search using that inverted index which returns a list of files that contain the query term / terms.
The search index must be in memory.

**Task 6**
Implement Sieve of Eratosthenes Algorithms for Prime Number.
In mathematics, the Sieve of Eratosthenes is a simple, ancient algorithm for finding all prime numbers up to any given limit.
Input: an integer n > 1.
Pseudocode, as follows

```
Let A be an array of Boolean values, indexed by integers 2 to n,
initially all set to true.

for i = 2, 3, 4, ..., not exceeding √n:
  if A[i] is true:
    for j = i2, i2+i, i2+2i, i2+3i, ..., not exceeding n:
      A[j] := false.

Output: all i such that A[i] is true.
```

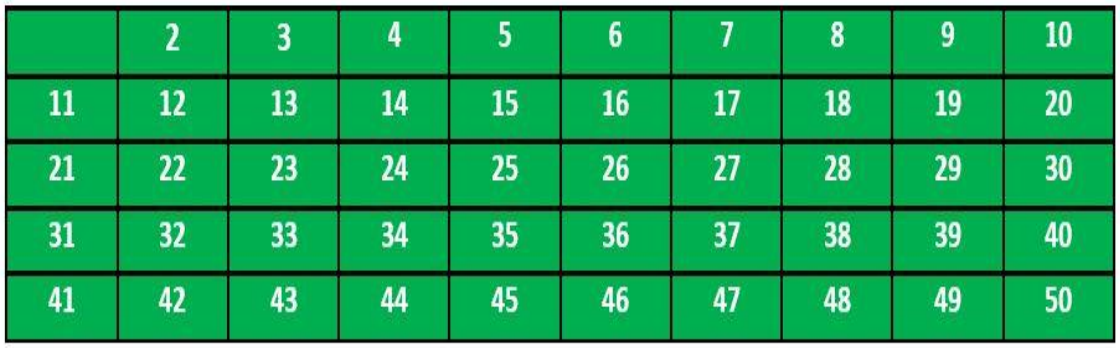
Following is the algorithm to find all the prime numbers less than or equal to a given integer *n* by Eratosthenes' method:

1. Create a list of consecutive integers from 2 to *n*: (2, 3, 4, …, *n*).
2. Initially, let *p* equal 2, the first prime number.

3. Starting from $p^2$, count up in increments of *p* and mark each of these numbers greater than or equal to $p^2$ itself in the list. These numbers will be *p(p+1)*, *p(p+2)*, *p(p+3)*, etc..
4. Find the first number greater than *p* in the list that is not marked. If there was no such number, stop. Otherwise, let *p* now equal this number (which is the next prime), and repeat from step 3.

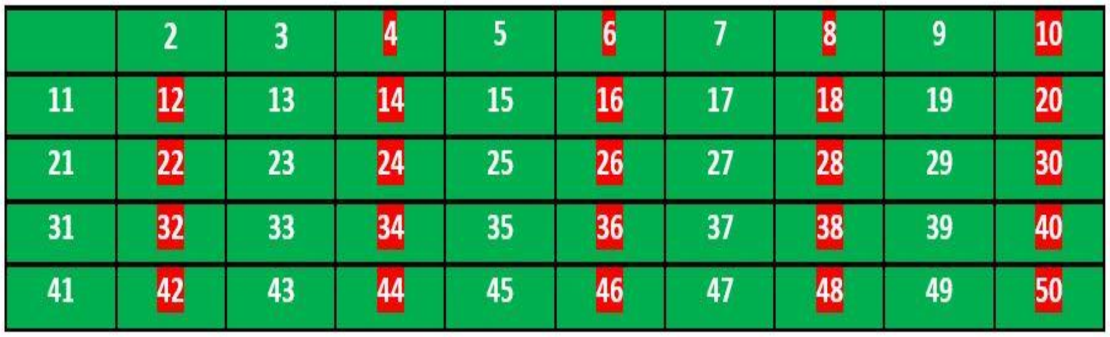
Below you will find the solution described with an example

We create a list of all numbers from 2 to 50.



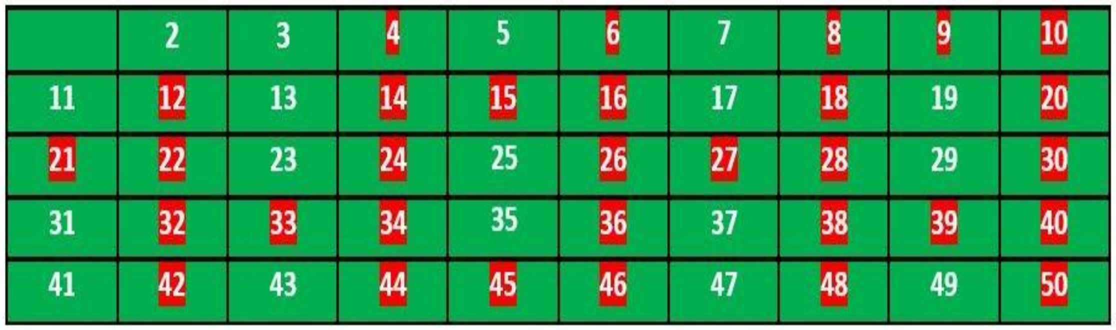
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |

According to the algorithm we will mark all the numbers which are divisible by 2 and are greater than or equal to the square of it.



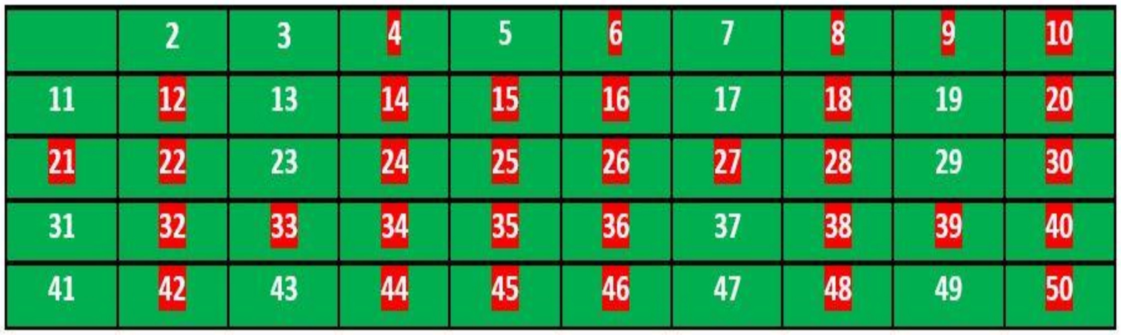
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | **4** | 5 | **6** | 7 | **8** | 9 | **10** |
| 11 | **12** | 13 | **14** | 15 | **16** | 17 | **18** | 19 | **20** |
| 21 | **22** | 23 | **24** | 25 | **26** | 27 | **28** | 29 | **30** |
| 31 | **32** | 33 | **34** | 35 | **36** | 37 | **38** | 39 | **40** |
| 41 | **42** | 43 | **44** | 45 | **46** | 47 | **48** | 49 | **50** |

Now we move to our next unmarked number 3 and mark all the numbers which are multiples of 3 and are greater than or equal to the square of it.



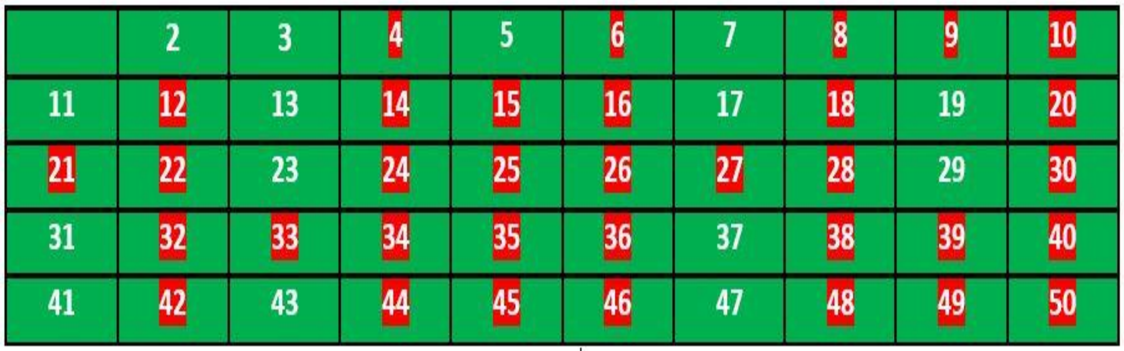
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | **4** | 5 | **6** | 7 | **8** | **9** | **10** |
| 11 | **12** | 13 | **14** | **15** | **16** | 17 | **18** | 19 | **20** |
| **21** | **22** | 23 | **24** | 25 | **26** | **27** | **28** | 29 | **30** |
| 31 | **32** | **33** | **34** | 35 | **36** | 37 | **38** | **39** | **40** |
| 41 | **42** | 43 | **44** | **45** | **46** | 47 | **48** | 49 | **50** |

We move to our next unmarked number 5 and mark all multiples of 5 and are greater than or equal to the square of it.



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |

We continue this process and our final table will look like below:



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |

So the prime numbers are the unmarked ones: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47.