

Contents

Basic Skills.....	2
Recursion.....	3
Recursion Problems	4
Numerical Techniques	5
Prime Checking.....	5
Greatest Common Divisor.....	5
Numerical Techniques Problems.....	6
Past Paper First Questions	7
Passwords (2000).....	7
Eenee, Meenee, Mainee, Mo! (2001).....	7
Lojban (2002)	7
ISBN (2003)	8
Anagrams (2006).....	8
Cards (2007).....	8
Goldbach Conjecture (2008).....	9
Digit Words (2009).....	9
Past Paper Question Hints	10
First Questions.....	10

Basic Skills

You need to be very comfortable with the basic features of your language. The code samples in this pack will be given as pseudocode, so you need to know how to do all of the following things in your language:

Basic Constructs

- If statements including the use of 'and', 'or' and 'not', and all of the comparison operators
- For and while loops, particularly how to make a for loop iterate over a range

Functions

- How to write a function, including how to accept as input and return lists
- How to access a global variable from within a function

Lists

- Making a list
- Iterating over a list
- Concatenating two lists (sticking them together)
- Removing and inserting elements from and to a list
- Extracting a sublist
- Counting the length of a list

Strings

- Making a string
- Concatenating strings
- Iterating over characters in a list
- Taking substrings
- Counting the length of a string

Numbers

- Basic arithmetic
- Finding the square root of a number
- Rounding a number to the nearest integer, or the next or previous one (ceiling and floor)
- Raising a number to a power
- How to do modulo ($a \bmod b$ is the remainder when a is divided by b)

Recursion

Recursion is a technique for solving problems by breaking a problem down into smaller versions of the same type of problem, until the problems are so small that they can be solved easily.

For example, imagine we are writing a function that outputs the letter “A”, n times in a row. We could easily do this with a loop:

```
function RowOfA( $n$ ):  
    result = ""  
    for  $i = 1$  to  $n$ :  
        result = result + "A"  
    return result
```

However, with many problems we come across there will not be an obvious way to solve the problem using loops, and we will have to solve the problem by breaking it down into smaller problems of the same type. We will do this for the problem above. To do this, we need to realise that a row of n “A”s is the same as a single “A” next to a row of $n - 1$ “A”s, and that a row of 0 “A”s is just an empty string. Now we can write a function that calls itself with smaller and smaller versions of the problem, until the problem is trivial to solve.

```
function RowOfA( $n$ ):  
    if  $n = 0$ :  
        return ""  
    else:  
        return "A" + RowOfA( $n - 1$ )
```

To understand this, let's imagine calling RowOfA(2), to get this: “AA”. When we call RowOfA(2), obviously $n = 2$ and so the condition on the if statement will come out false, meaning that the else part will be run. This makes a string starting with “A” and finishing with the string returned by RowOfA(1). The computer now leaves the execution of RowOfA(2) alone for a minute while it works out RowOfA(1). Again, the condition is false and so the else is run. The computer makes a string that will be the result of RowOfA(1), which consists of an “A” followed by the string returned by RowOfA(0). The computer now does the same as before, leaving the execution of RowOfA(1) while it works out RowOfA(0). This will return “”, an empty string. The computer now goes back to its half-executed RowOfA(1) which it can now work out as “A” + “”, which is just “A”. It returns this and the computer can now work out RowOfA(2), which will be “A” + “A”, which is “AA”, as we expected.

It would be useful to know the standard terminology used to talk about recursion. The size of problem that is so small we can just solve it straight away is called the ‘base case’. The breaking down of a large problem into smaller problems of the same type is called the ‘inductive step’. This type of problem solving is often known as ‘Divide and Conquer’, for obvious reasons.

To ensure that we understand recursion and how it is useful, we will look at another problem that can be solved using recursion.

This problem is to determine whether a string is a palindrome or not. A palindrome is a string that is the same when read forwards as it is when read backwards. For example: “12321”, “anna” and “ab1cd323dc1ba” are all palindromes, whereas “hello”, “1212” and “Anna” are not.

To do this problem with a loop is still manageable, but it is quite complicated compared to the first example in this section, and we are not going to do it here. On the other hand, the recursive solution is quick and easy, and it a good example of why recursion sometimes works best.

Firstly, we will decide on the base case – the size of problem that is trivially easy to solve. When the string has length 0 or it has length 1 then clearly it is the same backwards as forwards, and so we would always return true in this situation. We should now figure out what the inductive step is. A string is a palindrome if the first and last character match and the substring excluding the first and last character is a palindrome. We can now write our function:

```
function CheckP(str):
    if length(str) <= 1:
        return true
    else:
        if str.first = str.last and CheckP(str.substring(1, length(string) - 2)):
            return true
        else: return false
```

Hopefully you can see that this function keeps checking that the outside two characters of smaller and smaller strings are equal until it finds two that are not equal, in which case it returns false, or the string is so small that it can be declared a palindrome without any further work, in which case the function returns true.

These examples only included breaking the problem into one smaller problem, but the situations where recursion is most powerful are where the problem needs to be broken into more than one smaller problem. One of the problems below use this, and the most powerful sorting and searching algorithms which we will see later use this method too.

Recursion Problems

1. Write a function that takes a list of numbers as input and returns the largest number in the list, without using a loop.
2. The factorial of a number (an integer), written as $n!$ is the number multiplied by each of the numbers smaller than it, down to 1. For example: $4! = 4 \times 3 \times 2 \times 1 = 24$. Write a function without using a loop that takes an integer as input and returns the factorial of the number.
3. The Fibonacci sequence is defined like this: the first and second number in the sequence are both 1, and later numbers in the sequence are the sum of the previous two. (ie 1, 1, 2, 3, 5, 8...) Write a function without using a loop that takes a number n as input and returns the n th number in the Fibonacci sequence.

Numerical Techniques

There are techniques for working with numbers (mainly integers) that you should know in case they come up, since it may be difficult to work these out in the exam.

Prime Checking

Prime numbers are one of the most important parts of mathematics and come into computer science through their use in Public Key Cryptography, which protects the credit card details of online shoppers. For the BIO it is not necessary to understand how RSA works (!), but it is important to know how to check whether a number is a prime or not.

There are no clever ways to be certain whether a number is prime other than to try to divide it by all of the numbers that could possibly divide it and see if any of them leave a remainder. If they do, then the number is not prime. It is easy to simply run through all of the numbers smaller than the number we are checking, but there is no need. If a number is not prime (it is composite) then it is guaranteed to have at least one factor (one number that divides it without leaving a remainder) that is less than or equal to the square root of the number.

So we only have to check from 2, up to the square root of the number, rounded down, for any remainders. The function for rounding down is usually called `floor()`, but might be called `int()` or `truncate()`, depending on the language. There is no harm in rounding to the nearest integer, since the worst that can happen is that you will make one extra division every time you check a number. The way to find the remainder is with the modulo function, where $a \bmod b$ means the remainder when a is divided by b . We also need to check for 2 and 1, which are prime and composite, respectively, since the loop will not cover these numbers correctly. The function is therefore:

```
function CheckPrime( $n$ ):
    if  $n = 1$ : return false
    if  $n = 2$ : return true
    for  $i = 2$  to  $\text{floor}(\sqrt{n})$ :
        if  $n \bmod i \neq 0$ :
            return false
    return true
```

This algorithm is not very fast for large numbers, but you shouldn't have to check very large numbers for primality since the mathematics involved in this is very complicated.

Greatest Common Divisor

The greatest common divisor (or highest common factor) of a pair of numbers is the largest number that divides both of them without leaving a remainder. For example, the greatest common divisor of 25 and 15 is 5. There is a reasonably quick way to do this called the Euclidean Algorithm. All you have to do is keep subtracting the smaller number of the pair from the bigger (swapping over when you need to) until both are equal, and then return the number that they are both equal to. For example, to find $\text{gcd}(25, 15)$ we start by subtracting 15 from 25, which leaves us with (10, 15). Then we subtract 10 from 15 which leaves us with (10, 5). Finally we subtract 5 from 10 which leaves us with (5, 5), and so $\text{gcd}(25, 15) = 5$.

The code for this is not at all difficult; we just need to take the case where each number is larger separately:

```
function gcd(a, b):  
    while a ≠ b:  
        if a > b:  
            a = a - b  
        else:  
            b = b - a  
    return a
```

Finding the greatest common divisor of more than two numbers works like this:

$$\text{gcd}(a, b, c) = \text{gcd}(a, \text{gcd}(b, c))$$

So to find the greatest common divisor of a list of numbers, you find $\text{gcd}(x, y)$ where x is the first element of the list, and y is the greatest common divisor of the rest of the list. This naturally leads to a recursive solution, although an iterative solution is not significantly more difficult to understand or implement.

Numerical Techniques Problems

1. Write a function that takes a number greater than 2 as input and returns the largest prime that is smaller than that number.
2. Write a function that takes a pair of numbers as input and returns whether the greatest common divisor of those numbers is prime.
3. Write a function that takes a list of numbers and returns the greatest common divisor of those numbers.

Past Paper First Questions

Passwords (2000)

Some computer programs try to prevent easily guessable passwords being used, by rejecting those that do not meet some simple rules. One simple rule is to reject passwords which contain any sequence of letters immediately followed by the same sequence.

For example:

APPLE	Rejected	(P is repeated)
CUCUMBER	Rejected	(CU is repeated)
ONION	Accepted	(ON is not immediately repeated)
APRICOT	Accepted	

Write a program which inputs a password (between 1 and 10 upper case letters) and then prints Rejected if the password is rejected, or Accepted otherwise. Your program should then terminate.

[22 marks]

Sample input:
BANANA
Sample output:
Rejected

Eenee, Meenee, Mainee, Mo! (2001)

A group of friends are standing in a circle, playing a game. They proceed around the circle, chanting a rhyme and pointing at the next person (clockwise) on each word. When the rhyme finishes whoever is being pointed at leaves the circle. They then start again, pointing where they left off, and the game continues until only one person is left.

For example, suppose there are six friends (numbered 1 to 6) standing around the circle and the rhyme is "Eenee, Meenee, Mainee, Mo!". The first person to leave the circle is number 4, followed by 2 then 1, 3 and 6. Number 5 is left.

Write a program that inputs two numbers (each between 1 and 1000 inclusive), the first being n , the number of friends, and the second the number of words in the rhyme. The output should be the number of the last person left.

[24 marks]

Sample input:
7 3
Sample output:
4

You should assume that the friends are numbered (clockwise) from 1 to n , and that the count begins at person 1.

Lojban (2002)

Counting in Lojban, an artificial language developed over the last forty years, is easier than in most languages. The numbers from zero to nine are:

1	2	3	4	5	6	7	8	9	0
pa	re	ci	vo	mu	xa	ze	bi	so	no

Larger numbers are created by gluing the digits together. For example, 123 is "pareci".

Write a program that reads in a Lojban string (representing a number less than or equal to 1,000,000) and outputs it in numbers.

[22 marks]

Sample input:
renonore
Sample output:
2002

ISBN (2003)

An ISBN (International Standard Book Number) is a ten digit code which uniquely identifies a book. The first nine digits represent the book and the last digit is used to make sure the ISBN is correct. To verify an ISBN you calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third all the way until you add 1 times the last digit. If the final number leaves no remainder when divided by 11 the code is a valid ISBN.

For example, 0201103311 is a valid ISBN, since:

$$10 \times 0 + 9 \times 2 + 8 \times 0 + 7 \times 1 + 6 \times 1 + 5 \times 0 + 4 \times 3 + 3 \times 3 + 2 \times 1 + 1 \times 1 = 55$$

Each of the first nine digits can take a value between 0 and 9. Sometimes it is necessary to make the last digit equal to ten; this is done by writing the last digit as X. For example, 156881111X.

Write a program that reads in a valid ISBN with a single missing digit, marked with a ?, and outputs the correct value for the missing digit.

[24 marks]

Sample input:

15688?111X

Sample output:

1

Anagrams (2006)

Two words are anagrams of each other if they can both be formed by rearranging the same combination of letters. For example, GADGET and TAGGED are anagrams because they both contain one occurrence of each of the letters A, D, E and T, and two occurrences of the letter G.

Write a program which inputs two words (each of which will contain fewer than 10 uppercase letters) and then prints Anagrams if they are anagrams of each other, or prints Not anagrams otherwise. Your program should then terminate.

[24 marks]

Sample input:

SUMMER

RESUME

Sample output:

Not anagrams

Cards (2007)

A card game is played with a deck of forty cards, containing each of the numbers from 1 to 10 exactly four times. The game is scored according to the following two rules: a point is given for each pair of cards with identical numerical values and for any group of cards whose numerical values sum to 15.

For example, the set of cards 8, 8 and 8 is worth three points since there are three different pairs of cards with identical numerical values. The set 10, 5, 2 and 3 is worth two points since there are two groups of cards whose numerical values sum to 15.

Write a program which inputs 5 numbers (each of which will be between 1 and 10 inclusive) indicating the numerical values on 5 different cards. Your program should print out the number of points these 5 cards are worth and then terminate.

[24 marks]

Sample input:

3 3 3 2 10

Sample output:

6

Goldbach Conjecture (2008)

It is known that all even numbers between 4 and 300,000,000,000,000 are equal to the sum of two primes (a fact that is believed to be true for all larger even numbers as well, and called the Goldbach Conjecture).

For example, $30 = 7 + 23$. There are two other ways of expressing 30 as the sum of two primes, which are $11 + 19$ and $13 + 17$. These are the only ways of expressing 30 as the sum of two primes, since the order of the numbers in the additions does not matter.

Write a program which inputs a single even number (between 4 and 10,000 inclusive) and outputs a single number which is the number of different ways the input can be expressed as the sum of two primes.

[25 marks]

Sample input:

22

Sample output:

3

Digit Words (2009)

A digit word is a word where, after possibly removing some letters, you are left with one of the single digits: ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT or NINE.

For example:

- BOUNCE and ANNQUNCE are digit words, since they contain the digit ONE.
- ENCODE is not a digit word, even though it contains an O, N and E, since they are not in order

Write a program which reads in a single upper-case word (with at most fifteen letters) and determines if it is a digit word.

If the word is not a digit word you should output the word NO. If the word is a digit word you should output the digit it contains, as a number.

You will not be given any words which contain more than one digit.

[25 marks]

Sample input:

BOUNCE

Sample output:

1

Sample input:

ENCODE

Sample output:

NO

Past Paper Question Hints

First Questions

Passwords

Solve for the problem for just repeated sequences of two characters first. Can you see how to solve the full problem?

Eenee, Meenee, Mainee, Mo!

You need to keep track of who is still in the circle, and who is currently being pointed at.

Lojban

Treat each two character string separately.

ISBN

There are only ten possibilities; you need to check which is right.

Anagrams

For each letter in the first word, you need to find a corresponding one in the second word, but make sure you don't count the same letter twice.

Cards

The problem size is small so try all possibilities. For the groups summing to 15 part, remember that for each card there two possibilities: to include it in the group you are checking and to not include it.

Goldbach Conjecture

Loop through the different ways to add two numbers together, checking if both are prime numbers.

Digit Words

To check whether a word contains a digit, you need to look along the word for each of the letters of the digit in order.