Project 1

Task 1

1) Introduction to the problem

The first problem aims to create a random positive integer. The number of the digits composing the integer is the input. The output should be presented in a string containing all the digits of the randomly generated positive integer/useful when working with large numbers/.

2) Problem Solution

Let's assume the input is n. The general idea for solving this question contains several steps:

- The first step is to consider the use of the built-in function *randi*, which creates a random integer from the uniform distribution in an interval we specify.
- The next step is to create n random numbers from 0 to 9 by the above function, which are the digits a_1 to a_n of a positive integer.
- Consider also that the first digit a_1 should not be 0, since then the number of digits becomes $n - 1$.[1]
- Collect the digits in a vector and convert it to a string.

3) Code Implementation

The problems described in the section "*2) Problem Solution*" are solved using the code shown in Figure 7 in the Appendix.

On line 1 is defined the function, which is called "myrandi". On lines 6-11 is generated a "for loop" which goes through all the digits of the integer (a_1 to a_k) and assigns them random values from 0 to 9. A random integer from 1 to 9 is chosen for the first digit because it should be different from 0 /Line 10/. On Line 14 the results of every iteration of the loop are collected in a vector c. On Line 16 we convert the numeric vector to a string. On line 18 the space between the different digits in the string is eliminated.

4) Results

To validate the code we choose $n = 15$. The output from the code in Figure 7 is shown in Figure 1.

```
1  >> myrandi(15)
2  ans =
3
4      '420441186552454'
```

Figure 1: Command window output showing the digits of a random positive integer in a string

Indeed the output is a string, containing 15 digits with random numbers from 0 to 9.

[1] We can add infinitely many zeros in front of 1 as 00001, but in an algebraic point of view that is still 1 with just one digit.

Task 2

1) Background history

Vedic mathematics dates back to the ancient Indian history. "The origin of Vedic mathematics were derived from the Sanskrit word 'Veda' which means 'Knowledge' and it contains list of 16 sutras to solve mathematical calculations in easy and faster way. Urdhva Tiryagbhyam is the most popular sutra and it follows the 'vertical and crosswise' multiplication and it reduces greatly the number of partial products." (Rodda Srinivas, 2019, p.44-50)

To make it clearer an example of how the algorithm works is given.

Multiplication of two 2-digit Integers

Let us multiply 14×21.

1. Firstly, we will present them one under the other:

$$\begin{array}{cc} 1 & 4 \\ 2 & 1 \\ \hline \end{array}$$

2. Secondly, we multiply /"vertically"/ the last digit of the first number with the last digit of the second number to get $4 \times 1 = 4$ so we put 4 as the last digit under the line.

$$\begin{array}{cc} 1 & 4 \\ 2 & 1 \\ \hline & 4 \end{array}$$

3. Thirdly, we multiply /"crosswise"/ the first digit from the first number with the second digit from the second number $(1 \times 1 = 1)$ and the second digit from the first number with the first digit from the second number $(4 \times 2 = 8)$ and add the two products together - $1 + 8 = 9$, so we write 9 as a second digit under the line.

$$\begin{array}{cc} 1 & 4 \\ 2 & 1 \\ \hline & 9 \quad 4 \end{array}$$

4. Finally, we multiply the first digits of the two integers, which in our case is $1 \times 2 = 2$, so we write 2 as the first digit under the line.

$$\begin{array}{cc} 1 & 4 \\ 2 & 1 \\ \hline 2 & 9 \quad 4 \end{array}$$

Therefore, our answer is $14 \times 21 = 294$

Proof:

The first integer can be written as $(ax + b)$, where $x = 10$.
Similarly, the second integer can be written as $(cx + d)$, where $x = 10$.

$$(ax + b) \times (cx + d) = (ac) \cdot x^2 + (ad) \cdot x + (bc) \cdot x + bd$$
$$(ax + b) \times (cx + d) = (ac) \cdot x^2 + (ad + bc) \cdot x + bd$$

Therefore, a visualisation of the number we get is:

$$ac|ad + bc|bd$$

(Vertically and crosswise, Part 1, 2008)

2) Method

The proof above is one method in which the Urdhva-Tiryagbhyam algorithm can be generalised for multiplying two $n-$digit numbers.

Let $A = a_1a_2...a_n$ be our first integer, containing n digits $a_1, a_2, ..., a_n$ and $B = b_1b_2...b_n$ be our second integer, containing also n digits $b_1, b_2, ..., b_n$. A good approach is to convert A and B into polynomials in the following way /as in the proof for 2-digit integers/:

$$A = a_1 \times x^{n-1} + a_2 \times x^{n-2} + ... + a_n \tag{1}$$

(2)

Then, by multiplying equations 1 and , we get the following polynomial P:

$$P = a_1b_1 \times x^{2n-2} + (a_1b_2 + a_2b_1) \times x^{2n-3} + ... + a_nb_n \tag{3}$$

If the coefficients of every x of the polynomial are collected, then they create the digits of the number obtained after the multiplication, which are:

a_1b_1	$a_1b_2 + a_2b_1$	...	a_nb_n

Table 1: Coefficients of the polynomial.

However, that is not the case when some of the coefficients are larger than 9 because we have to deal with the carries then. That is why we find the modulo 10 of all the digits /the reminder obtained after dividing the digit by 10/ and the integer part of the division by 10. We will use the following table:

coefficient	1	2	...	k
modulo				
integer part				

Table 2: Finding the modulo and integer part for the coefficients of the polynomial.

, where k is the number of coefficients of the polynomial. So this results in:

coefficient	1	2	...	k
	a_1b_1	$a_1b_2 + a_2b_1$	...	a_nb_n
	fix(1,10)	mod(1,10)+fix(2,10)	mod(k-1,10)+fix(k,10)	mod(k,10)

Table 3: The algorithm for computing the digits of the result.

, where fix(k,10) is the integer part of dividing the k-th coefficient by 10 and mod(k,10) is the remainder of dividing the k-th coefficient by 10. The calculations in the last row are repeated until there is no number larger than 9. When that is true, the desired result is obtained and these are the digits of the integer, arranged from first to last digit.

Task 3

1) Introduction to the problem

In this question is presented the implementation of the algorithm from Task 2. The input required is two integers - a and b in a string form. Compared to Task 2, here they can have

a different number of digits. The output is the multiplication of the two integers, using Vedic Mathematics.

2) Problem Solution

Let's assume a has n digits and b has m digits. If we present the integers in the same form as equations *1* and , but with n substituted by m for the integer b, then we get the polynomial in equation *3* with $a_n b_n$ substituted by $a_n b_m$. After these alterations, the rest of the question remains the same since it works with the coefficients of the polynomial. Using the last row from Table *3* until all the numbers are less than or equal to 9, it is then possible to find the digits of the integer.

3) Code Implementation

The code for this task is shown in Figure 9. On lines 10 to 17, we are encoding the polynomials in equations *1* and with n substituted by m for the integer b. In line 18 we are finding the equation *3* with $a_n b_n$ substituted by $a_n b_m$. In 19 we are creating a vector similar to Table *1*. In line 23 we are setting up a while loop to continue with the iterations until all the numbers in the vector are less than or equal to 9. In lines 24 to 27, we are doing Table *2*. In lines 30 to 33, we are doing the calculations from the last row of Table *3*.

We will validate our code in Task 4, in Section "*4) Results*".

Task 4

1) Introduction to the problem

By using the functions we created - **myrandi** and **vedicmultiply**, the task here is to find the product of two random integers - one of length 30 and one of length 40.

2) Problem Solution

The problem solution is not challenging, since we have already created the required functions. First, two random integers a /with 30 digits/ and b /with 40 digits/ are created using **myrandi**, and second, their product is calculated using **vedicmultiply**.

3) Code

The code which is used is the one in Figure *2*:

```
1 a=myrandi(30); %creating a random integer a with 30 digits
2 b=myrandi(40); %creating a random integer b with 30 digits
3 vedicmultiply(a,b)%finding the product of a and b
```

Figure 2: MATLAB code for multiplying a and b

4) Results

The output of the code is presented in Figure *3*.

```
>> Q4
a =
    '459681849500587698401805620134'
b =
    '602543661908141614532687598541456605475 1'
ans =
    "27697838491079135042342343424463998143587193830721571059037313519566 34"
```

Figure 3: Command window output for multiplying a and b

To validate the results, the same integers a and b are used, but this time their product is calculated by using the built-in standard multiplication in MATLAB, shown in Figure 4.

```
a=sym('459681849500587698401805620134');
b=sym('602543661908141614532687598541456605475 1');
a*b
```

Figure 4: MATLAB code for validation

The output, which is consistent with the output from our function, is visible in Figure 5.

```
>> check
ans =
27697838491079135042342343424463998143587193830721571059037313519566 34
```

Figure 5: Command window output for validation

Task 5

1) Introduction to the problem

This task requires creating a plot, where the x-axis is the number of digits of an integer and the y-axis is the average time to do a multiplication. Let our two integers be a and b and let N be the number of their digits /from 10 to 50 with step 10/.

2) Method and Code

The code for this task is presented in Figure 8. The solution to the problem consists of the following steps: First, we initialise a vector in which the time values will be stored /Line 1/. Then we loop over the values of N /Line 2 to 13/. Inside the loop, the two integers a and b are assigned /Line 5 and 6/. Multiply them 100 times /Line 4 and 7/. The result obtained is divided by 100 to establish the average time for which a multiplication is performed /Line 11/. Furthermore, the resulting 5 time values are stored in the vector T which represents the y-axis /Line 12/. Another vector has been created with the number of digits of the numbers N, representing the x-axis of the plot /Line 15/. Finally, the desired graph is drawn up /Line 16/.

3) Results

After running the code, shown in Figure 8 in the Appendix, the output is the plot in Figure 6:

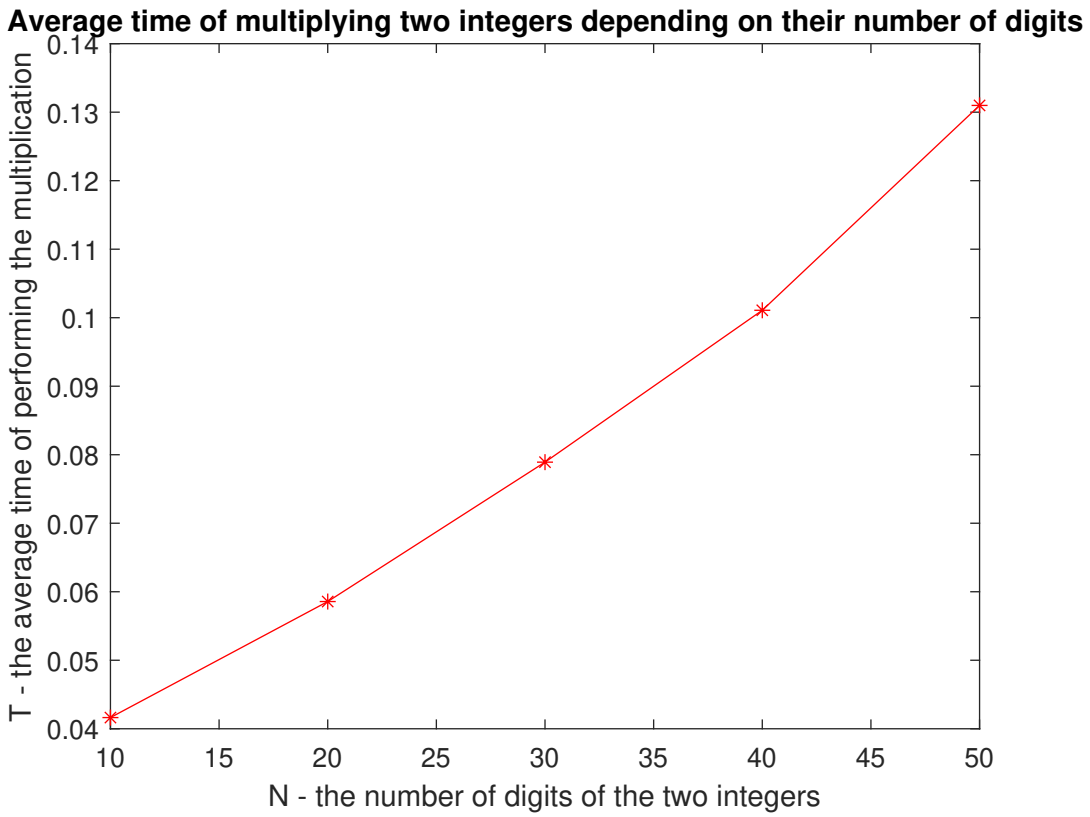

Figure 6: Average time of multiplying two integers depending on their number of digits

Figure 6 shows that the computation time increases as the number of digits of the numbers we multiply increases. However, times are low and are approximately in the interval [0.04, 0.13].

References

Rodda Srinivas, Feb 2019 - "Implementation of an Efficient Vedic Multiplier Incorporating Parallel Prefix Adder Using 'Urdhva Tiryagbhyam' Sutra", "International Research Journal of Innovations in Engineering and Technology"; Dharmapuri Vol. 3, Iss. 2: p. 44-50, accessed 19 October 2023, < https://www.proquest.com/docview/2608093813?pq-origsite=gscholar&fromopenview=true >.

Jul 2008 - "Vertically and crosswise", Part 1, Multiplication of two 2-Digit Numbers Algebraic Proof: Multiplication of two 2-digit numbers, accessed 20 October 2023, < https://liberius1776.wordpress.com/2008/07/26/vertically-and-crosswise-part-1/ >.

Appendix

Code listings

[All the codes have been tested in MATLAB.]

```
function [myans] = myrandi(n)
% The function generates a positive integer of length n
% The input should be a positive integer
% The ouput is a string and it contains the digits of the integer

% Create a for loop for k from 1 to n with step 1
for k=1:1:n
    % Generating random integer values from 0 to 9 for a_1 to a_k
    a(k)=randi([0,9]);
    % The first digit should not be zero
    a(1)=randi([1,9]);
end

% Show the output from all the iterations of the loop
c= a(1:k);
% Convert the output to a string containing a_1 to a_k
myans=num2str(c);
% Eliminate the space between the digits in the string
myans=strrep(myans, ' ', '');
end
```

Figure 7: Task 1: MATLAB code to find a random positive number of length n

```
T=[]; %initilasing a vector
for N=10:10:50 %for N from 10 to 50 digits with step 10
t=0; %setting t to zero to store the outputs from the for loop
for k=1:100 %for doing 100 multiplications
    a=myrandi(N);
    b=myrandi(N);
    tic;vedicmultiply(a,b);
    t=t+toc;% store the time for doing 100 multiplications of two integers
    %with number of digits from 10 to 50 with step 10
end
av=t/100; %find the average time to do one multiplication of every t from above
T=[T,av];%store all these average times in a vector
end
T;
N=[10:10:50];
plot(N,T,'r-*')
title('Average time of multiplying two integers depending on their number of digits');
xlabel('N - the number of digits of the two integers') ;
ylabel('T - the average time of performing the multiplication') ;
print -depsc2 myfig.eps
```

Figure 8: Task 5: MATLAB code for finding the average time to do a multiplication between two integers, each with number of digits from 10 to 50 with step 10.

```
function [myans] = vedicmultiply (a, b)
% vedicmultiply computes the product a*b
% a and b are input as strings
% myans is a list containing all the digits of the answer

n=length(a); %defining length of a
m=length(b); %defining length of b
x=sym('x');%defining a symbolic variable x to use for the polynomials
%Generating the polynomial version of the first number by for loop from its first to last digit
firstn=0;
for k=1:n
    firstn=firstn+a(k)*(x^(n-k));
end
secondn=0;
for i=1:m
    secondn=secondn+b(i)*(x^(m-i));%same for the second number
end
p=expand(firstn*secondn);%multiplying out the polynomials resulting in another one
c=sym2poly(p);%collecting the coefficient in a vector from the first one to the last one
d=length(c);%assigning a value to the length of the vector

%Creating a while loop for repeating a calculation until all of the numbers in the vector becomes less than or equal to 9
while any(c>9)
for z=1:d
   reminder(z)=mod(c(z),10);%Calculating the reminder of all the coeff in c after dividing by 10
   quotient(z)=fix(c(z)/10);%Calculating the integer part of all the coeff in c after dividing by 10
end

d=d+1;%Adding additional space for the reminder of the last digit when dividing by 10
carries =reminder(1:(z-1))+quotient(2:z);%adding the reminder and the integer part to deal with the carries
carries(end+1)=reminder(z);%putting modulo 10 of the last digit on last place in the vector
c1=[quotient(1), carries]; %putting the integer part of the first digit when dividing by 10 on the first place in the vector
c=c1(1:z+1);%updating the vector with the digits of the output needed
end
myans=num2str(c); %converting the vector to string
myans=strrep(myans, ' ', ''); %removing the space between the digits in the string
myans=str2sym(myans); %convert to symbolic to remove possible zeros at the beginning of the string
myans=string(myans);%convert back to string
end
```

Figure 9: Task 3: MATLAB code to multiply two integers using Vedic Mathematics