重 庆 大 学

学 生 作 业 报 告

课程名称 数学实验

组员 1 姓 名 马梓恒 学 号 20233124

组员 2 姓 名 周宏仰 学 号 20232647

组员 3 姓 名 李宇聪 学 号 20232137

开 课 时 间 2024 至 2025 学年第 一 学期

总 成 绩	

数 统 学 院 制

开 课 学 院 、 实 验 室 ： 航 空 航 天 学 院

实验时间 ： **2024** 年 **9** 月 **16** 日

课程名称	数学实验	实验项目名称	**MATLAB** 初步	实验项目类型				
				验证	演示	综合	设计	其他
指导教师	肖剑	成绩		✓				

题目 1

用 MATLAB 软件计算 1 道极限问题或积分问题。

$$\int_0^1 \left(x^2+3x+2\right)dx = \left[\frac{x^3}{3}+\frac{3x^2}{2}+2x\right]_0^1 = \frac{23}{6}$$

程序

```
syms x;
result = int(x^2 + 3*x + 2, x, 0, 1);
disp(result);
```

结果

$$\frac{23}{6}$$

分析

首先，代码通过 syms x 定义了一个符号变量 x，接着利用 int 函数计算多项式 $\int_0^1 \left(x^2+3x+2\right)dx = \left[\frac{x^3}{3}+\frac{3x^2}{2}+2x\right]_0^1 = \frac{23}{6}$ 在区间 [0, 1] 上的定积分。这个多项式的定积分相当于求该函数在指定区间内的面积。最后，disp(result) 将计算出的积分结果输出到命令窗口。核心思路是通过符号运算方式计算定积分，以得到特定区间内函数的累积值。

程序

```
numbers = [111, 1111, 11111];

for i = 1:length(numbers)
    num = numbers(i);
    fprintf('寻找自然数因数乘积为 %d 的组合:\n', num);
        for factor1 = 2:floor(sqrt(num))
        if mod(num, factor1) == 0
            factor2 = num / factor1;
            if factor2 ~= 1
                fprintf('%d = %d * %d\n', num, factor1, factor2);
            end
        end
    end
    fprintf('\n');
end
```

结果

寻找自然数因数乘积为 111 的组合:
111=3*37
寻找自然数因数乘积为 1111 的组合:

1111=11*101
寻找自然数因数乘积为 11111 的组合:
11111 = 41 * 271

分析

这段代码首先定义了一个包含多个目标数字的数组 numbers，接着通过循环逐个处理这些数字，并在每次迭代中打印出当前数字。然后，代码确定因数的搜索范围，从 2 遍历到当前数字的平方根，以寻找可能的因数。在检查每个因数时，使用 mod 函数判断当前数字能否被该因数整除，如果能，则记录下这个因数。之后，代码计算另一个因数，即将当前数字除以找到的因数，并确保这个因数不为 1，最后以可读的格式输出有效的因数组合。整体思路是找出每个目标数字的自然数因数组合，并将结果整齐地显示出来。

题目 3

用牛顿迭代法求方程 $x^2-2=0$ 的一个根，牛顿迭代法可以百度其公式，迭代的终止条件为前后两次求出的 x 的差的绝对值小于 10^{-5}.

程序

```
x = 1;
tolerance = 1e-5;
diff = 1;
max_iterations = 100;
iteration = 0;
while diff > tolerance && iteration < max_iterations
    x_new = x - (x^2 - 2) / (2 * x);
    diff = abs(x_new - x);
    x = x_new;
    iteration = iteration + 1;
end

fprintf('经过 %d 次迭代，方程的一个根为: %.6f\n', iteration, x);
```

结果

经过 4 次迭代，方程的一个根为: 1.414214

分析

首先，初始化了一些参数，其中 **x = 1** 是初始猜测值，**tolerance = 1e-5** 是设定的终止条件，确保迭代在两次猜测值差异足够小时停止。此外，**max_iterations** 设置最大迭代次数为 **100**，以防止无限循环，并用 **iteration** 记录当前的迭代次数。
接着，进入迭代过程的 **while** 循环，循环的条件是当前差值大于容忍度，并且迭代次数未达到上限。在循环内部，使用牛顿迭代公式 $x_{\text{new}}=x-\frac{x^2-2}{2x}$ 计算新的猜测值。随后，计算前后两次猜测值的差 **diff**，并用新值更新 **x**。每次迭代完成后，迭代次数加一。

循环结束后，使用 fprintf 函数输出迭代次数和计算得到的根，格式化为小数点后六位。这段代码的最终目的是找出 22 的数值解。

选做题

某公司生产一种特殊的密码锁，密码锁的密码由 5 个数字组成，这些数字从集合{1, 2, 3, ..., 6}中选取，并且必须满足以下条件：
（1）至少有三个数字是互不相同的。
（2）任意相邻两个数字之差的绝对值不为 3。
所有满足这些条件的密码组合被视为一个批次中的不同密码锁。求一个批次中共有多少把这样的密码锁？

程序

```
function count = optional1()
    digits_set = [1, 2, 3, 4, 5, 6];

    all_combinations = combvec(digits_set, digits_set, digits_set, digits_set,
digits_set)';

    count = 0;

    for i = 1:size(all_combinations, 1)
        password = all_combinations(i, :);
        if length(unique(password)) < 3
            continue;
        end

        valid = true;
        for j = 1:length(password)-1
            if abs(password(j) - password(j+1)) == 3
                valid = false;
                break;
            end
        end

        if valid
            count = count + 1;
        end
    end
end
```

结果

```
Ans =3384
```

分析

这段代码实现计算满足特定条件的密码组合数量。首先，函数定义了一组可用的数字集，包括从 1 到 6

的数字。接着，它使用 `combvec` 函数生成所有可能的长度为 5 的数字组合，并将结果转置以形成每一行代表一个潜在密码组合的矩阵。接下来，函数初始化一个计数器用于记录符合条件的密码数量。随后，它通过遍历所有组合来处理每一行，检查每个组合。如果组合中不同数字的数量少于 3，代码会跳过该组合。然后，函数通过一个内部循环检查相邻数字之间的绝对差是否等于 3，若是，便将一个标志设置为 `false` 并跳出循环。最后，如果组合满足所有条件，计数器便会增加。这段代码的最终目的是返回有效密码组合的数量，适用于安全密码生成或组合数量的计算分析。