Contents	

Part 1:

Procedural Programming, Pascal

What is Pascal?

Pascal is a general purpose high level language that was originally developed by Nicklaus Wirth in the early 1970s.

It was developed for teaching programming as a systematic discipline and to develop reliable and efficient programs, it is easy to understand and maintain the Pascal programs.

The computer programming languages C and Pascal have similar times of origin, influences, and purposes. Both were used to design and compile.

Both are descendants of the ALGOL language series.

ALGOL introduced

programming language support for structured programming, where programs are constructed of single entry and single exit constructs such as if, while, for and case.

Comparing Pascal grammar and C grammar:

The computer programming languages C and Pascal have similar times of origin, influences, and purposes. Both were used to design and compile. Both are descendants of the ALGOL language series. ALGOL introduced programming language support for structured programming, where programs are constructed of single entry and single exit constructs such as if, while, for and case.

Pascal vs C

Readability:

Pascal: Generally more readable due to its strong typing and clear syntax, making it easier for beginners to understand.

C: More concise but can be less readable due to its flexibility and use of pointers, which may confuse new programmers.

Writability:

Pascal: Encourages structured programming with a focus on clarity, which can lead to easier maintenance.

C: Offers more freedom and power for low-level programming, allowing for more efficient code, but can lead to complex and less maintainable code.

Reliability:

Pascal: Strong typing and error-checking features enhance reliability and reduce runtime errors.

C: Offers less built-in error checking, making it easier to introduce bugs, but experienced programmers can manage this effectively.

Cost:

Pascal: Generally lower cost for development in educational contexts due to its simplicity, but limited in terms of libraries and system-level programming.

C: Higher initial investment in terms of complexity but often more cost-effective for large-scale or system-level projects due to its performance and extensive libraries.

Pascal vs C in syntax:

Comments:

Pascal: { block comments } single- line comment

(* block comments *) multi- line comment

C: // single- line comment

/* block comments */ multi- line comment

Declaration of variable:

Pascal: var

variable_list : type; x : integer;

C: type variable_list;

int x;

Operators and Expression:

Pascal: s := (a + b)

if (a and b)

C: s = a + b

if (a && b)

For loop:

Pascal:

for < variable-name > := <

initial_value > to [down to] <

final_value > do

EX : for a := 10 to 50 do

C:

for (initializationStatement ; condition;updateStateent)

{ // statements inside the body of loop }

EX : for (i = 1; i < 50; ++i) {}

Print:

Pascal: writeLn('……..');

C: printf("%d", ...);

Program skeleton in Pascal:

```
1  program SkeletonProgram; { The name of the program }
2
3  { Variable declarations }
4  var
5     x: Integer;
6
7  { Function declarations }
8  Function DoSomething;
9  begin
10    { Placeholder for Function logic }
11 end;
12
13 { Main program block }
14 begin
15    { Initialization code }
16
17    { Call Functions }
18    DoSomething;
19
20    { Additional main program logic }
21
22    { End of program }
23 end.
24
```

Pascal Control Statements:

While loop:

```
while (condition) do
begin
{statement to execute while
the condition is true }
end;
```

For loop:

```
for counter := initial_value to
final_value do
begin
{ statement to execute for a
specified number of iterations }
end;
```

Repeat-until loop:

```
{repeat-until loop syntax:}

program {Name};
begin
   {initial value};
      repeat
      {statements}
   until {condition};
end.
```

If statement:

```
if condition then
begin
  { statements to execute if condition is true }
end
else
begin
  { statements to execute if condition is false }
end;
```

Case statement:

```
case expression of
value1:{code block};
value2:{code block};
{…}
end;
```

1- Area of the circle

A program that calculates the area of a circle using user-provided radius and π, displaying the result formatted to two decimal places.

```
program CircleArea;

const
  PI = 3.14159;

var
  radius, area: real;

begin
  write('Enter the radius of the circle: ');
  readln(radius);

  area := PI * radius * radius;

  writeln('The area of the circle is: ', area:0:2);
end.
```

```
15 lines compiled, 0.0 sec
Enter the radius of the circle: 5
The area of the circle is: 78.54
```

2- Prime numbers

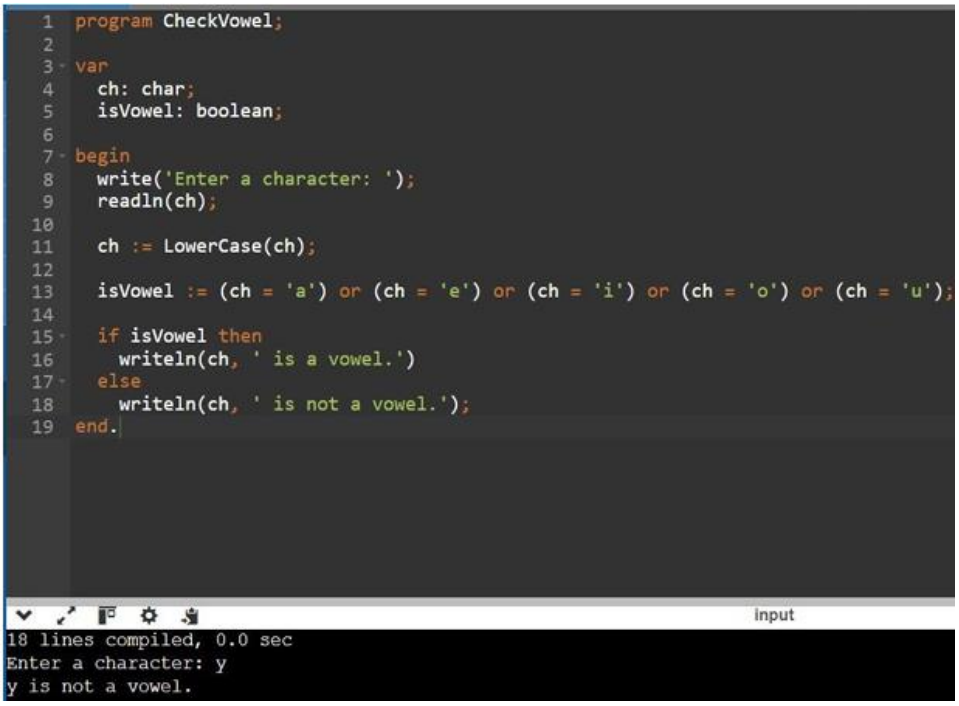
This program checks if a positive integer is prime by testing divisibility and displays the result.

```
program CheckPrime;

var
  number, i: integer;
  isPrime: boolean;

begin
  write('Enter a number: ');
  readln(number);

  isPrime := (number > 1);

  for i := 2 to trunc(sqrt(number)) do
    if (isPrime and (number mod i = 0)) then
      isPrime := false;

  if isPrime then
    writeln(number, ' is a prime number.')
  else
    writeln(number, ' is not a prime number.');

end.
```

```
Enter a number: 7
7 is a prime number.
```

3- Vowel character

This program prompts the user to enter a character, checks if it is a vowel (a, e, i, o, u), and displays the result.

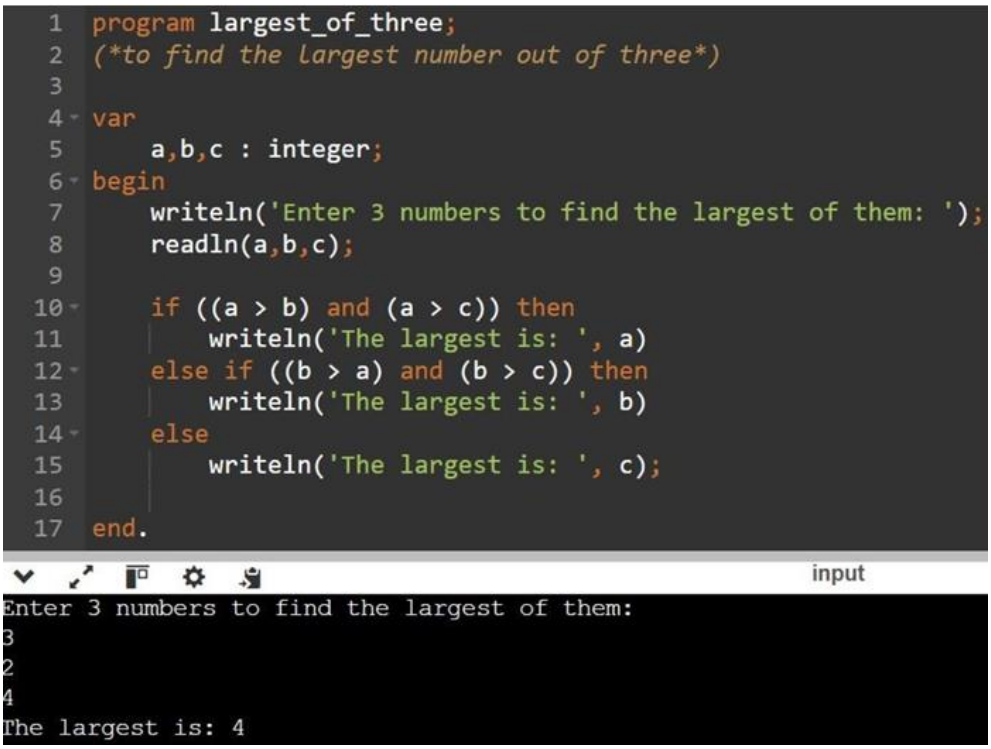


```
program CheckVowel;

var
  ch: char;
  isVowel: boolean;

begin
  write('Enter a character: ');
  readln(ch);

  ch := LowerCase(ch);

  isVowel := (ch = 'a') or (ch = 'e') or (ch = 'i') or (ch = 'o') or (ch = 'u');

  if isVowel then
    writeln(ch, ' is a vowel.')
  else
    writeln(ch, ' is not a vowel.');
end.
```

```
input
18 lines compiled, 0.0 sec
Enter a character: y
y is not a vowel.
```

4- The largest number out of 3

This program comparing between 3 numbers entered by user then displays the largest number of them.



```
program largest_of_three;
(*to find the largest number out of three*)

var
    a,b,c : integer;
begin
    writeln('Enter 3 numbers to find the largest of them: ');
    readln(a,b,c);

    if ((a > b) and (a > c)) then
        writeln('The largest is: ', a)
    else if ((b > a) and (b > c)) then
        writeln('The largest is: ', b)
    else
        writeln('The largest is: ', c);

end.
```

```
input
Enter 3 numbers to find the largest of them:
3
2
4
The largest is: 4
```

5- Word counter

This program count words in a statement entered by user then displays the number of the words.

```
program wordCount;
(*count the number of etered words by user*)

var
    words : string;
    count, i : integer;

begin
    writeln('Enter a statement: ');
    readln(words);

    count := 1;

    for i:=1 to length(words) do
        if words[i] = ' ' then
            count := count + 1;

    writeln('The number of words are: ', count);
end.
```

input

```
Enter a statement:
Welcome to Pascal
The number of words are: 3
```

6- Multiplication table

This program displays the multiplication table of a number that entered by user.

```
program multiplication_Table;
(*show the multiplication table of the enterd number by user*)

var
    number : integer;
    i : integer;
begin
    writeln('Enter a number: ');
    readln(number);

    for i:=1 to 12 do
        writeln( i ,' x ', number, ' = ', i*number);
end.
```

input

```
Enter a number:
5
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
6 x 5 = 30
7 x 5 = 35
8 x 5 = 40
9 x 5 = 45
10 x 5 = 50
11 x 5 = 55
12 x 5 = 60
```

7- Squared Array

This Pascal program efficiently squares each number in a list. It takes a list of numbers and squares each one, simplifying calculations where squared values are needed.

```pascal
program SquareArray;
const
MAX_SIZE = 100;
type
IntArray = array[1..MAX_SIZE] of Integer;
var
arr: IntArray;
size, i: Integer;
begin
Write('Enter the size of the array: ');
Read(size);
Write('Enter ', size, ' elements: ');
for i := 1 to size do
Read(arr[i]);

arr[i] := arr[i] * arr[i];
Write('Squared array:');
for i := 1 to size do
Write(' ', arr[i]);
end.
```

```
Enter the size of the array: 1
Enter 1 elements: 2
Squared array: 4
```

8- Calculate the tangent line

This Pascal program calculates the value of (y) on a tangent line given an initial point (x_0, y_0) and the slope (m), for a specified (x) value, using the formula for the equation of a line: (y = y_0 + m * (x - x_0).

```pascal
program TL ;
var y,y0,x,x0,m : real ;
begin
Write ('Enter the value of x0 ') ;
ReadLn(x0);
Write ('Enter the value of y0 ') ;
ReadLn(y0);
Write ('Enter the slope of m ') ;
ReadLn(m);
Write('Enter the value of x for which you want to find y: ');
ReadLn(x);
y := y0 + m * ( x - x0 ) ;
WriteLn('The value of y at x = ', x:0:2, ' is: ', y:0:2);
end.
```

```
Enter the value of x0 2
Enter the value of y0 3
Enter the slope of m 2
Enter the value of x for which you want to find y: 4
The value of y at x = 4.00 is: 7.00
```

9- Month Name

This Pascal code is a simple program that takes a user-inputted month number (1-12) and outputs the corresponding month name.

```
program MonthName;
var
monthNumber: integer;
begin
writeln('Enter a month number (1-12): ');
readln(monthNumber);
case monthNumber of
1: writeln('January');
2: writeln('February');
3: writeln('March');

5: writeln('May');

7: writeln('July');

9: writeln('September');
10: writeln('October');
11: writeln('November');
12: writeln('December');
else
writeln('Invalid month number');
end;
end.
```

```
Enter a month number (1-12):
8
August
```

10- Calculating the factorial number

A program that calculates the product of all positive integers up to a given number. For example, the factorial of 5 is 5 * 4 * 3 * 2 * 1 = 120.

```
program FactorailNum;

var num ,fact , i : integer;
 begin
 fact :=1;
 write('Write integer number:');
 readln(num);
 if num < 0 then
 writeln('Error no factorail for nagativ number.') else
 begin
 for i := 1 to num do
 fact := fact * i;
 writeln('The factorail number for ' , num ,'is:', fact);
 end;
 end.
```

```
14 lines compiled, 0.0 sec
Write integer number:3
The factorail number for 3is:6
```

11- Simple calculator

A simple calculator program allows the user to perform basic arithmetic operations (addition, subtraction, multiplication, and division) between two numbers. The program first asks the user to enter two numbers, then, it asks the user to select an operation (+, -, *, or /). Based on the selected operation, the program performs the appropriate calculation using conditional statements (if-else).

```
program SimpleCalc;
var
num1 , num2 , result: real;
operation: char;
begin
write('inter the first number:');
readln(num1);
write('inter the operation (+,-,*,/):');
readln(operation);
write('inter the second number:');
readln(num2);
if operation = '+' then
result := num1 + num2
else if operation = '-' then
result := num1 - num2
else if operation = '*' then
result := num1 * num2
else if (operation = '/') and (num2 <> 0) then
result := num1 / num2
else
begin
writeln('Invalid oprator or division by zero.');
exit;
end;
writeln('The result is: ', result:0:2);
end.
```

input

```
Linking a.out
26 lines compiled, 0.1 sec
inter the first number:5
inter the operation (+,-,*,/):+
inter the second number:5
The result is: 10.00
```

12- Temperature conversion program

This program converts temperatures between Celsius and Fahrenheit. The user is prompted to select a conversion type: 1.Celsius to Fahrenheit 2.Fahrenheit to Celsius. Based on the choice, the program reads the input temperature and converts it using the appropriate formula: -Celsius to Fahrenheit: F = (C * 9/5) + 32 -Fahrenheit to Celsius: C = (F - 32) * 5/9 ,finally, the program prints the converted temperature.

```
program TemperatureConverting;
var
temperature , convtemp: real;
number: integer;

function CelToFah(cel: real):real;
begin
CelToFah := (cel * 9 / 5) + 32;
end;
function FahToCel(fah: real): real;
begin
FahToCel := (fah - 32) * 5 / 9;
end;
begin;
writeln('Select conversion type:');
writeln('1:from celsius to fahrenheit-');
writeln('2:from fahrenheit to celsius-');
write('inter your choise: 1 or 2:');
readln(number);
case number of 1:begin
write('inter Temperature in celsius:');
readln(temperature);
convtemp := CelToFah(temperature);
writeln('temperature in fahrenheit is:', convtemp:0:2);
end;
```

```
writeln('temperature in fahrenheit is:', convtemp:0:2);
end;
2: begin
write('inter Temperature in fahrenheit:');
readln(temperature);
convtemp := FahToCel(temperature);
writeln('temperature in celsius is:',convtemp:0:2);
end;
else
writeln('wrong choice');
end;
end
```

input

```
Free Pascal Compiler version 3.2.2+dfsg-9ubuntu1 [2022/04/11] for x86_64
Copyright (c) 1993-2021 by Florian Klaempfl and others
Target OS: Linux for x86-64
Compiling main.pas
Linking a.out
35 lines compiled, 0.1 sec
Select conversion type:
1:from celsius to fahrenheit-
2:from fahrenheit to celsius-
inter your choise: 1 or 2:  1
inter Temperature in celsius:  0
temperature in fahrenheit is:32.00
```

13- Simple password validator

This program allows the user to enter a password which then the program checks if the password is valid according to the conditions. Conditions for a valid password are: Should have at least one number. Should have at least one uppercase letter. Should be more or equal to 8 characters long.

```
program SimplePasswordValidator;
var
  password: string;
  i: integer;
  hasUpper, hasDigit: boolean;
begin
  writeln('Simple Password Validator');
  write('Enter your password: ');
  readln(password);
  hasUpper := false;
  hasDigit := false;
  if Length(password) < 8 then
  begin
    writeln('Password must be at least 8 characters long.');
  end
```

```
  else
  begin
    for i := 1 to Length(password) do
    begin
      if password[i] in ['A'..'Z'] then
        hasUpper := true
      else if password[i] in ['0'..'9'] then
        hasDigit := true;
    end;
    if hasUpper and hasDigit then
      writeln('Password is valid.')
    else
      writeln('Password must contain at least one uppercase letter and one digit.');
  end;
end.
```

```
29 lines compiled, 0.0 sec
Simple Password Validator
Enter your password: password
Password must contain at least one uppercase letter and one digit.
```

14- Binary to decimal converter

A program that converts a binary number into its decimal equivalent. It allows the user to enter a binary number, processes each bit from right to left, and calculates the corresponding decimal value by adding the powers of 2 for each '1' encountered. The final decimal result is then displayed to the user.

```
program BinaryToDecimalConverter;
var
  binaryStr: string;
  decimalValue, i, power: integer;
begin
  writeln('Binary to Decimal Converter');
  write('Enter a binary number: ');
  readln(binaryStr);
  decimalValue := 0;
  power := 1;
  for i := Length(binaryStr) downto 1 do
  begin
    if binaryStr[i] = '1' then
      decimalValue := decimalValue + power;
    power := power * 2;
  end;
  writeln('The decimal value is: ', decimalValue);
end.
```

```
25 lines compiled, 0.0 sec
Binary to Decimal Converter
Enter a binary number: 101
The decimal value is: 5
```

15- Simple math problem generator

A program that creates basic arithmetic problems by randomly selecting two integers (1 to 100) and one of four operations: addition, subtraction, multiplication, or division (with checks to avoid division by zero). Users solve the problem and receive instant feedback on their answer, including the correct solution if needed.

```
program SimpleMathProblemGenerator;
var
  num1, num2, result: integer;
  operator1: char;
  userAnswer: integer;
begin
  Randomize;
  num1 := Random(100) + 1;
  num2 := Random(100) + 1;
  case Random(4) of
    0: begin
      operator1 := '+';
      result := num1 + num2;
    end;
    1: begin
      operator1 := '-';
      result := num1 - num2;
    end;
    2: begin
      operator1 := '*';
      result := num1 * num2;
    end;
    3: begin
      operator1 := '/';
      num2 := Random(99) + 1;
      result := num1 div num2;
    end;
  end;
  writeln('Solve the following problem:');
  writeln(num1, ' ', operator1, ' ', num2, ' = ?');
  write('Your answer: ');
  readln(userAnswer);

  if userAnswer = result then
    writeln('Correct!')
  else
    writeln('Incorrect. The correct answer is ', result);

end.
```

```
39 lines compiled, 0.1 sec
Solve the following problem:
90 + 25 = ?
Your answer: 33
Incorrect. The correct answer is 115
```

16- Sorting

This program implements the insertion sort algorithm to sort an array of integers. It defines an array of 5 elements, initially set to {10, 20, 30, 40, 50}. The sorting process moves through the array, shifting elements that are larger than the current key to the right, and placing the key in its correct position. The program prints the array before and after sorting.

```
program Sorting;

Var x: Integer;
Var arr : array[1..5] of Integer;

procedure Sort(size : Integer );
Var i, j, key : Integer;
Begin
    For i := 2 to size do
    Begin
        key := arr[i];
        j := i;
        While ((j > 1) AND (arr[j-1] > key)) do
        Begin
            arr[j] := arr[j-1];
            j := j - 1;
        End;
        arr[j] := key;
    End;
End;

Begin

    arr[1] := 10;
    arr[2] := 20;
    arr[3] := 30;
    arr[4] := 40;
    arr[5] :=50;

    writeln(' unsorted ');
    for x:= 1 to 5 do
    write(' ', arr[x] );

    Sort(5);
    writeln();
    writeln(' sorted ');

    for x:= 1 to 5 do
    write(' ', arr[x] );

End.
```

```
unsorted
10 20 30 40 50
sorted
10 20 30 40 50
```

17- Countries

This program prompts the user to select one of three countries (Egypt, Jordan, or Bahrain) and displays details about the chosen country, such as distance from Riyadh, flight duration, capital city, climate, and main airport. If the user enters an invalid number, the program shows an error message.

```
program CountriesInfo;
var
    Number: integer;

function CheckCountries(num: integer): integer;
begin
    if(num = 1) then
        writeln('Egypt:'
        +#13#10+'Distance from Riyadh - 1200 KM'
        +#13#10+'Flight duration - 2 hours and 30 minutes'
        +#13#10+'Capital city - Cairo'
        +#13#10+'Climate - hot desert climate with mild winters'
        +#13#10+'Airport - Cairo International Airport')
    else if (num = 2) then
        writeln('Jordan:'
        +#13#10+'Distance from Riyadh - 900 KM'
        +#13#10+'Flight duration - 1 hour and 45 minutes'
        +#13#10+'Capital city - Amman'
        +#13#10+'Climate - Mediterranean climate with hot summers'
        +#13#10+'Airport - Queen Alia International Airport')
    else if(num = 3) then
        writeln('Bahrain:'
        +#13#10+'Distance from Riyadh - 400 KM'
        +#13#10+'Flight duration - 1 hour'
        +#13#10+'Capital city - Manama'
        +#13#10+'Climate - hot, humid summers and mild winters'
        +#13#10+'Airport - Bahrain International Airport')
    else
        writeln('Wrong entry!');
end;

begin
    writeln('Welcome!');
    writeln('Choose one of these countries:');
    writeln('1- Egypt');
    writeln('2- Jordan');
    writeln('3- Bahrain');
    readln(Number);
    CheckCountries(Number);
end.
```

```
Welcome!
Choose one of these countries:
1 - Egypt
2 - Jordan
3 - Bahrain
2
Jordan:
Distance from Riyadh - 1,315 kilometers
Flight duration - Approximately 2 hours
Capital city - Amman
Climate - Moderate climate with hot summers and cold winters
Airport - Queen Alia International Airport
```

18- Even and odd numbers

This program asks the user to enter six numbers, then displays them. It separates and prints the even and odd numbers based on a simple modulus check (mod 2). The program concludes with a thank-you message.

```
program Even_Odd;
var
    i: integer;
    num: array[1..6] of integer;
begin
    writeln();
    writeln('Hello, please enter any 6 numbers: ');
    for i := 1 to 6 do
    begin
        readln(num[i]);
    end;
    writeln('Original numbers: ');
    for i := 1 to 6 do
    begin
        write(num[i], ' ');
    end;
    writeln();
    writeln('Even numbers: ');
    for i := 1 to 6 do
    begin
        if (num[i] mod 2 = 0) then
            write(num[i], ' ');
    end;
    writeln();
    writeln('Odd numbers: ');
    for i := 1 to 6 do
    begin
        if (num[i] mod 2 <> 0) then
            write(num[i], ' ');
    end;
    writeln();
    writeln('Thank you!');
end.

```

```
Hello, please enter any 6 numbers:
1
2
3
4
5
6
Original numbers:
1 2 3 4 5 6
Even numbers:
2 4 6
Odd numbers:
1 3 5
Thank you!
```

Part 2:

Functional programming, Scheme

What's Scheme?

Scheme is a functional programming language derived from Lisp. It is simple, minimalistic, and designed for teaching programming concepts, handling symbolic computations, and creating small to medium-sized applications. Scheme emphasizes recursion, higher-order functions, and clean, expressive syntax, making it popular in academic and research settings.

Program skeleton in Scheme:

```
; Define a variable
(define x 0)

; Define a function
(define (do-something)
   ; Placeholder for function logic
)

; Main program block
(begin
   ; Initialization code

   ; Call functions
   (do-something)

   ; Additional main program logic
)

```

1-Area of the circle

This Scheme program calculates the area of a circle by prompting the user for the radius, computing the area using π×r^2, and displaying the result rounded to two decimal places.

```
(define PI 3.14159)

(define (circle-area)
  (display "Enter the radius of the circle: ")
  (let ((radius (read)))
    (let ((area (* PI radius radius)))
      (display "The area of the circle is: ")
      (display (format "~,2f" area))
      (newline))))

(circle-area)
```

```
Enter the radius of the circle: 5
The area of the circle is: 78.54
```

2-Prime numbers

The program checks divisibility for numbers up to the square root and then outputs whether the number is prime.

```
(define (check-prime)
  (display "Enter a number: ")
  (let* ((number (read))
         (isPrime (> number 1)))
    (define (is-divisible n i)
      (= (modulo n i) 0))

    (define (check-prime-loop n i)
      (if (<= i (sqrt n))
          (if (is-divisible n i)
              (set! isPrime #f)
              (check-prime-loop n (+ i 1)))))

    (check-prime-loop number 2)

    (if isPrime
        (display (string-append (number->string number) " is a prime number.\n"))
        (display (string-append (number->string number) " is not a prime number.\n")))))

(check-prime)
```

```
Run
Enter a number: 20
20 is not a prime number.
```

3-Vowel character

This program prompts the user to enter a character, checks if it is a vowel (a, e, i, o, u), and displays the result.

```
(define (is-vowel? ch)
  (or (char=? ch #\a)
      (char=? ch #\e)
      (char=? ch #\i)
      (char=? ch #\o)
      (char=? ch #\u)
      (char=? ch #\A)
      (char=? ch #\E)
      (char=? ch #\I)
      (char=? ch #\O)
      (char=? ch #\U)))

(define (check-vowel)
  (display "Enter a character: ")
  (newline)
  (let ((ch (read-char)))
    (if (is-vowel? (char-downcase ch))
        (display (string-append (string ch) " is a vowel.\n"))
        (display (string-append (string ch) " is not a vowel.\n")))))

(check-vowel)
```

```
Enter a character:
A
A is a vowel.
GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundation, Inc.

Guile comes with ABSOLUTELY NO WARRANTY; for details type `,show
 w'.
This program is free software, and you are welcome to redistribu
te it
under certain conditions; type `,show c' for details.

Enter `,help' for help.
>
```

4-The largest number out of 3

This program asks the user to input three numbers then determines the largest among them using a conditional structure, after that displays the largest number to the user.

```
; Largest of three numbers program

(define (largest-of-three a b c)
  (cond
    ((and (>= a b) (>= a c)) a)
    ((and (>= b a) (>= b c)) b)
    (else c)))

(display "Enter 3 numbers: ")
(newline)
(let ((a (read)) (b (read)) (c (read)))
  (display "The largest number is: ")
  (display (largest-of-three a b c))
  (newline))
(newline)
```

```
Run

Enter 3 numbers:
15 30 4
The largest number is: 30
```

5-Word counter

This program reads a sentence from the user and counts the number of words by checking for spaces between characters, then displays the total word count based on the input.

```
; Word Counter program

(define (word-count str)
  (define (count-words s count)
    (if (null? (cdr s))
        count
        (count-words (cdr s) (if (char=? #\Space (car s)) (+ count 1) count))))
  (count-words (string->list str) 1))

(display "Enter a sentence: ")
  (newline)
(let ((input (read-line (current-input-port))))
  (display "Number of words: ")
  (display (word-count input))
  (newline))
(newline)
```

```
Run

Enter a sentence:
Sana Aledilbi
Number of words: 2
```

6-Multiplication table

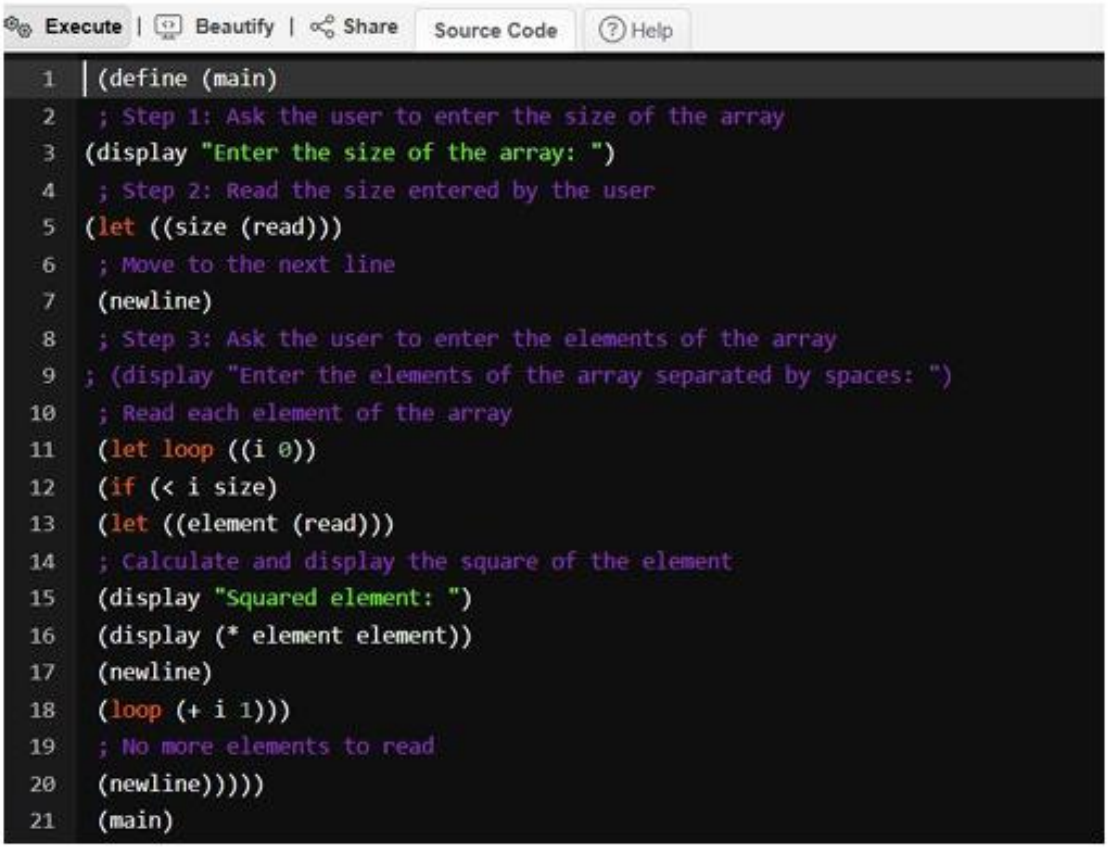
This program asks the user to enter a number to print the multiplication table for that number, from 1 to 12, then generates and displays each row of the table in the format "num * i = result."

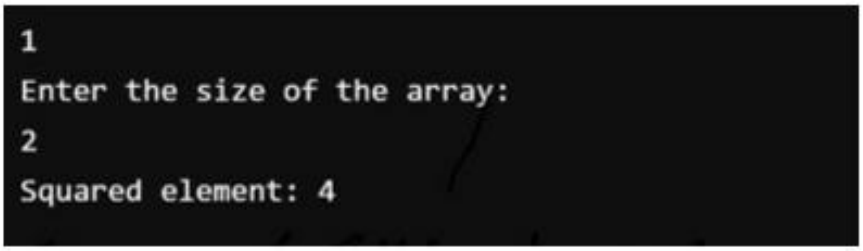
```
3   ; Multiplication Table program
4
5   (define (print-multiplication-table num)
6     (define (print-row i)
7       (when (<= i 12)
8         (display (string-append (number->string num) " x "
    (number->string i) " = " (number->string (* num i))))
9         (newline)
10        (print-row (+ i 1))))
11    (print-row 1))
12
13  (display "Enter a number: ")
14    (newline)
15  (let ((number (read)))
16    (print-multiplication-table number))
17  (newline)
18
```

```
Run
Enter a number:
6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
6 x 11 = 66
6 x 12 = 72
```

7-Squared Array

This code prompts the user to enter the size of an array, reads the size, then prompts the user to enter elements of the array one by one. For each entered element, it calculates and displays its square.





8- Calculate the tangent line

This Scheme program calculates the value of (y) on a tangent line given an initial point (x_0, y_0) and the slope (m), for a specified (x) value, using the formula for the equation of a line: (y = y_0 + m * (x - x_0)

Execute | Beautify | Share Source Code Help

```
(define (tangentLine)
(display "Enter the value of x0: ")
(newline)
(let ((x0 (read)))
(display "Enter the value of y0: ")
(newline)
(let ((y0 (read)))
(display "Enter the slope (m): ")
(newline)
(let ((m (read)))
(display "Enter the value of x for which you want to find y: ")
(newline)
(let ((x (read)))
(let ((y (+ y0 (* m (- x x0)))))
(display "The value of y at x = ")
(display x)
(display " is: ")
(display y)
(newline)))))))
(tangentLine)
```

```
Enter the value of x0:
2
Enter the value of y0:
3
Enter the slope (m):
2
Enter the value of x for which you want to find y:
4
The value of y at x = 4 is: 7
```

9-Month Name

This scheme code is a simple program named "MonthName" that takes a user inputted month number (1-12) and outputs the corresponding month name. with error handling for invalid inputs.

```
(define (month-name)
(let loop ()
(display "Enter a month number (1-12): \n")
(let ((month (read)))
(cond ((and (integer? month) (>= month 1) (<= month
12))
(case month
((1) (display "January\n"))
((2) (display "February\n"))
((3) (display "March\n"))
((4) (display "April\n"))
((5) (display "May\n"))
((6) (display "June\n"))
((7) (display "July\n"))
((8) (display "August\n"))
((9) (display "September\n"))
((10) (display "October\n"))
((11) (display "November\n"))
((12) (display "December\n"))))
(else
(display "Invalid input.\n")
(loop))))))
(month-name)
```

```
Enter a month number (1-12):
5
May
```

10- Calculating the factorial number

A program that calculates the product of all positive integers up to a given number. For example, the factorial of 5 is 5 * 4 * 3 * 2 * 1 = 120.

main.scm

```
1  (define (fact n)
2  (if (< n 0)
3  (error "Error: no factorail for negative numbers" )
4  (fact-helper n 1)))
5  (define (fact-helper n acc)
6    (if (= n 0)
7        acc
8       (fact-helper (- n 1) (* acc n))))
9  (display "Enter an integer: ")
10 (newline)
11 (let ((n (read)))
12   (display "The factorial is:")
13   (newline)
14   (display (fact n))
15   (newline))
16
```

AI | Console | Shell

Show Only Latest | Clear History

Run | Ask AI

```
Enter an integer:
5
The factorial is:
120
GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundation,
Inc.

Guile comes with ABSOLUTELY NO WARRANTY; for detai
ls type `,show w'.
This program is free software, and you are welcome
 to redistribute it
under certain conditions; type `,show c' for detai
ls.

Enter `,help' for help.
>
```

11- Simple calculator

A simple calculator program allows the user to perform basic arithmetic operations (addition, subtraction, multiplication, and division) between two numbers.

```
(define (simpleCalc)
  (display "Enter the frist number: ")
  (newline)
  (let ((num1 (read))
  (num2 0)
  (operation #\space)
        (result 0))
    (display "Enter the operation (+, -, *, /): ")
    (newline)
    (set! operation (read))
    (display "Enter the second number: ")
    (newline)
    (set! num2 (read))
    (if (equal? operation '+)
        (set! result (+ num1 num2))
    (if (equal? operation '-)
        (set! result (- num1 num2))
    (if (equal? operation '*)
        (set! result (* num1 num2))
    (if (and(equal? operation '/) (not (= num2 0)))
        (set! result (/ num1 num2))
        (begin (display "Error operator or division
      by zero")
          (newline)
             (exit))))))
    (display "The result is: ")
    (display result)
    (newline)))
    (simpleCalc)
```

```
Enter the frist number:
5
Enter the operation (+, -, *, /):
+
Enter the second number:
3
The result is: 8
GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundati
on, Inc.

Guile comes with ABSOLUTELY NO WARRANTY; for d
etails type `,show w'.
This program is free software, and you are wel
come to redistribute it
under certain conditions; type `,show c' for d
etails.

Enter `,help' for help.
>
```

12- Temperature conversion program

This program converts temperatures between Celsius and Fahrenheit. The user is prompted to select a conversion type.

```
(define (CelToFah cel)
  (+ (* cel (/ 9 5)) 32))
(define (FahToCel fah)
  (* (- fah 32) (/ 5 9)))
(define (temp-conversion)
  (display "Enter your choise: ")
  (newline)
  (display "1: from celsius to fahrenheit")
  (newline)
  (display "2: from fahrenheit to celsius")
  (newline)
  (let ((choise (read)) (temp 0) (converted-temp 0))
    (case choise((1)
       (display "Enter temperatur in celsius: ")
        (newline)
    (set! temp (read))
    (set! converted-temp (CelToFah temp))
      (display "Temprautur in fahrenheit:")
        (display converted-temp))
     ((2)
      (display "Enter temperatur in fahrenheit: ")
      (newline)
      (set! temp (read))
      (set! converted-temp (FahToCel temp))
        (display "Temprautur in celsius:")
         (display converted-temp))
      (else (display "Invalid choise ")))
    (newline)))
(temp-conversion)
```

```
Enter your choise:
1: from celsius to fahrenheit
2: from fahrenheit to celsius
1
Enter temperatur in celsius:
0
Temprautur in fahrenheit:32
GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundation,
Inc.

Guile comes with ABSOLUTELY NO WARRANTY; for detai
ls type `,show w'.
This program is free software, and you are welcome
 to redistribute it
under certain conditions; type `,show c' for detai
ls.

Enter `,help' for help.
>
```

13- Simple password validator

This Scheme program is a simple password validator that checks if a password meets specific criteria. It prompts the user to enter a password and verifies that it is at least 8 characters long, contains at least one uppercase letter, and includes at least one digit. The program uses helper functions to identify uppercase letters and numeric characters. If the password meets all the requirements, it displays a message indicating that the password is valid; otherwise, it provides feedback on the specific criteria that were not met.

```
(define (simple-password-validator)
  (display "Simple Password Validator\n")
  (display "Enter your password: ")
  (let ((password (read))
        (has-upper? #f)
        (has-digit? #f))
    (if (< (string-length password) 8)
        (display "Password must be at least 8 characters long.\n")
        (begin
          (for-each
           (lambda (ch)
             (cond ((char-upper-case? ch) (set! has-upper? #t))
                   ((char-numeric? ch) (set! has-digit? #t))))
           (string->list password))
          (if (and has-upper? has-digit?)
              (display "Password is valid.\n")
              (display "Password must contain at least one uppercase letter and one digit.\n"))))))

(define (char-upper-case? ch)
  (and (char>=? ch #\A) (char<=? ch #\Z)))

(define (char-numeric? ch)
  (and (char>=? ch #\0) (char<=? ch #\9)))

(simple-password-validator)
```

```
Show Only Latest  Clear History
Run                Ask AI
Simple Password Validator
"password"
Password must contain at least one uppercase letter and
 one digit.
GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundation, Inc.

Guile comes with ABSOLUTELY NO WARRANTY; for details ty
pe `,show w'.
This program is free software, and you are welcome to r
edistribute it
under certain conditions; type `,show c' for details.

Enter `,help' for help.
>
```

14- Binary to decimal converter

This Scheme program converts a binary string to its decimal equivalent. It processes the binary digits from right to left, multiplying each digit by increasing powers of 2 and summing the results when the digit is 1. The program lets the user enter a binary string, then computes and displays the corresponding decimal value.

```
(define (binary-to-decimal binary-str)
  (define (process-binary lst power decimal-value)
    (if (null? lst)
        decimal-value
        (let ((current-digit (car lst)))
          (process-binary
           (cdr lst)
           (* power 2)
           (if (char=? current-digit #\1)
               (+ decimal-value power)
               decimal-value)))))

  (let ((binary-list (string->list binary-str)))
    (process-binary (reverse binary-list) 1 0)))
```

main.scm

```
(define (run-binary-to-decimal-converter)
  (display "Binary to Decimal Converter\n")
  (display "Enter a binary number (as a string in quotes): ")
  (let ((binary-str (read)))
    (display "The decimal value is: ")
    (display (binary-to-decimal binary-str))
    (newline)))

(run-binary-to-decimal-converter)
```

AI | Console | Shell

Show Only Latest | Clear History

Run | Ask AI

```
Binary to Decimal Converter
"10110"
The decimal value is: 22
GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundation, Inc.

Guile comes with ABSOLUTELY NO WARRANTY; for details ty
pe `,show w'.
This program is free software, and you are welcome to r
edistribute it
under certain conditions; type `,show c' for details.

Enter `,help' for help.
>
```

15- Simple math problem generator

This Scheme program generates a simple math problem using random numbers and basic operators. It first takes user input to seed the random number generator, then randomly selects two numbers and an operator (addition, subtraction, multiplication, or division). The user is prompted to solve the problem, and their answer is compared to the correct result.

```scheme
(define (mod a b)
  (- a (* b (quotient a b))))
(define (random-in-range min max)
  (+ min (random (+ 1 (- max min)))))

(define (randomize-based-on-input)
  (display "Enter any number to help randomize: ")
  (let ((seed-input (read)))
    (do ((i 0 (+ i 1)))
        ((= i (mod seed-input 10)))
      (random 100))))

(define (generate-math-problem)
  (define num1 (random-in-range 1 100))
  (define num2 (random-in-range 1 100))
  (define operator (random 4))
  (define result 0)

  (cond
    ((= operator 0)
     (set! result (+ num1 num2))
     (set! operator #\+))
    ((= operator 1)
     (set! result (- num1 num2))
     (set! operator #\-))
    ((= operator 2)
     (set! result (* num1 num2))
     (set! operator #\*))
    ((= operator 3)
     (set! num2 (random-in-range 1 99))
     (set! result (quotient num1 num2))
     (set! operator #\/)))

  (display "Solve the following problem:\n")
  (display num1)
  (display " ")
  (display operator)
  (display " ")
  (display num2)
  (display " = ?\n")
  (display "Your answer: ")
  (let ((user-answer (read)))
    (if (= user-answer result)
        (display "Correct!\n")
        (begin
          (display "Incorrect. The correct answer is ")
          (display result)
          (display "\n")))))

(display "Simple Math Problem Generator\n")
(randomize-based-on-input)
(generate-math-problem)
```

```
Simple Math Problem Generator
54
Solve the following problem:
70 * 46 = ?
23
Incorrect. The correct answer is 3220
GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundation, Inc.

Guile comes with ABSOLUTELY NO WARRANTY; for details ty
pe `,show w'.
This program is free software, and you are welcome to r
edistribute it
under certain conditions; type `,show c' for details.

Enter `,help' for help.
>
```

16- Sorting

This program implements the insertion sort algorithm to sort an array of integers. It defines an array of 5 elements, initially set to {10, 20, 30, 40, 50}. The sorting process moves through the array, shifting elements that are larger than the current key to the right, and placing the key in its correct position. The program prints the array before and after sorting.

```
(define (arrange lis1 lis2)
  (if (null? lis1)
      lis2
      (if (null? lis2)
          lis1
          (if (< (car lis1) (car lis2))
              (cons (car lis1) (arrange (cdr lis1) lis2))
              (cons (car lis2) (arrange (cdr lis2) lis1))))))

(define (sort x)
  (if (null? x)
      '()
      (arrange (list (car x)) (sort (cdr x)))))

(display " unsorted ")
(newline)
(define Original '(40 50 30 10 20))
(display Original)
(newline)

(display " sorted ")
(newline)
(display (sort Original))
(newline)
```

```
 unsorted
(40 50 30 10 20)
 sorted
(10 20 30 40 50)
GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundation, Inc.
```

17- Countries

This program prompts the user to select one of three countries (Egypt, Jordan, or Bahrain) and displays details about the chosen country, such as distance from Riyadh, flight duration, capital city, climate, and main airport. If the user enters an invalid number, the program shows an error message.

```
(define (CheckCountry value)
  (cond
    ((= value 1) (display "Egypt:
Distance from Riyadh - 1614 KM
Flight duration - 2 hours and 45 minutes
Capital city - Cairo
Climate - Hot desert climate
Airport - Cairo International Airport"))

    ((= value 2) (display "Jordan:
Distance from Riyadh - 1242 KM
Flight duration - 2 hours
Capital city - Amman
Climate - Mediterranean climate with a hot dry summer
Airport - Queen Alia International Airport"))

    ((= value 3) (display "Bahrain:
Distance from Riyadh - 420 KM
Flight duration - 1 hour
Capital city - Manama
Climate - Hot desert climate, humid summers
Airport - Bahrain International Airport"))

    (else (display "Wrong entry!"))
  )
)

(display "\nWelcome!\nChoose one of these countries:\n1- Egypt\n2- Jordan\n3- Bahrain\n")
(display "-->")
(define num (read))
(CheckCountry num)
```

```
Welcome!
Choose one of these countries:
1 - Egypt
2 - Jordan
3 - Bahrain
2
Jordan:
Distance from Riyadh - 1,315 kilometers
Flight duration - Approximately 2 hours
Capital city - Amman
Climate - Moderate climate with hot summers and cold winters
Airport - Queen Alia International Airport
```

18- Even and odd numbers

This program asks the user to enter six numbers, then displays them. It separates and prints the even and odd numbers based on a simple modulus check (mod 2). The program concludes with a thank-you message.

```
(define (CheckEven v1)
  (cond
    ((zero? v1) "")
    ((negative? v1) "")
    ((even? v1) (display v1) " ")
    (else "")))
(define (CheckOdd v2)
  (cond
    ((zero? v2) "")
    ((negative? v2) "")
    ((odd? v2) (display v2) " ")
    (else "")))
(display "Hello, please enter any 6 numbers:")
(newline)
(define value1 (read))
(define value2 (read))
(define value3 (read))
(define value4 (read))
(define value5 (read))
(define value6 (read))
(display "Original numbers:")
(newline)
(display value1)(display " ")
(display value2)(display " ")
(display value3)(display " ")
(display value4)(display " ")
(display value5)(display " ")
(display value6)
(newline)
(display "Even values are:")
(newline)
(display (CheckEven value1))
(display (CheckEven value2))
(display (CheckEven value3))
(display (CheckEven value4))
(display (CheckEven value5))
(display (CheckEven value6))
(newline)
```

```
Hello, please enter any 6 numbers:
1
2
3
4
5
6
Original numbers:
1 2 3 4 5 6
Even values are:
2 4 6
Odd values are:
1 3 5
Thank You!GNU Guile 3.0.8
Copyright (C) 1995-2021 Free Software Foundation, Inc.
```

Part 3:

Logic programming, Prolog

What is Prolog?

Prolog is a logical programming language commonly used in the field of artificial intelligence and natural language processing. In Prolog, relationships (facts and rules) are represented primarily.

Program skeleton in Prolog:

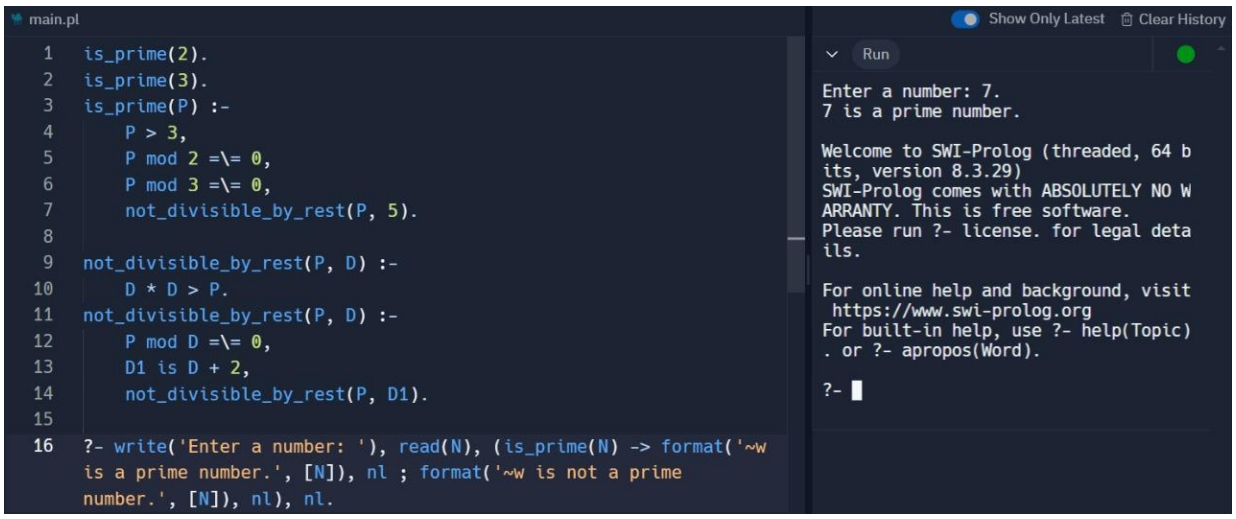
```
% Facts
fact1.
fact2.

% Rules
rule1(X) :- condition1(X).
rule2(Y) :- condition2(Y).

% Main predicate
main :-
    rule1(Result1), write(Result1), nl,
    rule2(Result2), write(Result2), nl.

% Queries to run:
% ?- main.          % Runs the main program.
% ?- rule1(X).      % Queries rule1.
% ?- rule2(Y).      % Queries rule2.

```

1-Area of the circle

A program that calculates the area of a circle using user-provided radius and π, displaying the result formatted to two decimal places.

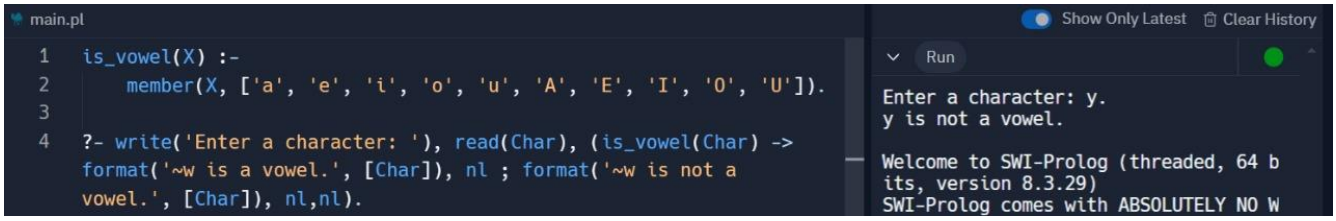
2-Prime numbers

This program checks if a positive integer is prime by testing divisibility and displays the result.



3-Vowel character

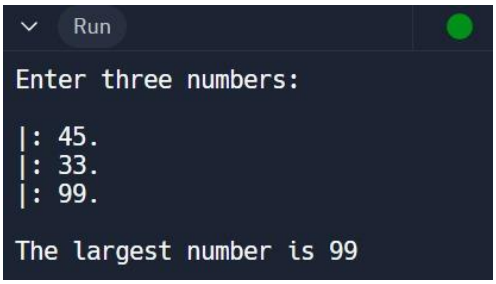
This program prompts the user to enter a character, checks if it is a vowel (a, e, i, o, u), and displays the result.



4-The largest number out of 3

This program comparing between 3 numbers entered by user then displays the largest number of them.

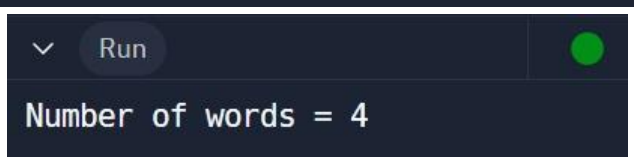
```
%Find_the_Largest_of_Three_Numbers

largest_of_three(A, B, C, Largest) :-
    (A >= B, A >= C -> Largest = A ;
     B >= A, B >= C -> Largest = B ;
     Largest = C).

find_largest :-
    writeln('Enter three numbers:'),nl,
    read(X),
    read(Y),
    read(Z),nl,
    largest_of_three(X, Y, Z, Largest),
    write('The largest number is '), writeln(Largest),nl.

?- find_largest.
```



5-Word counter

This program count words in a statement entered by user then displays the number of the words.

```
%Word_Count_Program

word_count(String, Count) :-
    split_string(String, " ", "", Words),
    length(Words, Count).

?- word_count("Hello world from Prolog", Count),
write("Number of words = "),write(Count), nl, nl.
```



6-Multiplication table

This program displays the multiplication table of a number that entered by user.

```
%Multiplication_Table_Program

print_multiplication_table(_, 13) :- !.
print_multiplication_table(Number, I) :-
    Format = '~w x ~w = ~w~n',
    format(Format, [Number, I, Number * I]),
    NextI is I + 1,
    print_multiplication_table(Number, NextI).

?- print_multiplication_table(5, 1),nl.

```

```
Run
5 x 1 = 5*1
5 x 2 = 5*2
5 x 3 = 5*3
5 x 4 = 5*4
5 x 5 = 5*5
5 x 6 = 5*6
5 x 7 = 5*7
5 x 8 = 5*8
5 x 9 = 5*9
5 x 10 = 5*10
5 x 11 = 5*11
5 x 12 = 5*12
```

7-Squared Array

This program efficiently squares each number in a list. It takes a list of numbers and squares each one, simplifying calculations where squared values are needed.

```
% Predicate to read a list of integers and print their squares
main :-
write('Enter the size of the array: '),
read_integer(Size),
calculate_squared_elements(Size, []).
% Predicate to read an integer from input
read_integer(X) :-
read_line_to_codes(user_input, Codes),
string_to_atom(Codes, Atom),
atom_number(Atom, X).
% Predicate to calculate the square of an element
calculate_squared_elements(0, SquaredList) :-
write('Squared elements: '), writeln(SquaredList).
calculate_squared_elements(Size, SquaredList) :-
Size > 0,
write('Enter element: '),
read_integer(Element),
Squared is Element * Element,
write('Squared element: '), writeln(Squared),
Size1 is Size - 1,
calculate_squared_elements(Size1, [Squared|SquaredList]).
:- main.
```

```
Enter the size of the array: 1
Enter element: 2
Squared element: 4
Squared elements: [4]
```

8- Calculate the tangent line

This program calculates the value of (y) on a tangent line given an initial point (x_0, y_0) and the slope (m), for a specified (x) value, using the formula for the equation of a line: (y = y_0 + m * (x - x_0).

```
tangent_line :-
write('Enter the value of x0: '), read_line_to_codes(user_input, X0_input),
atom_codes(X0_atom, X0_input),
(X0_atom == stop ->
halt;
(atom_number(X0_atom, X0) ->
read_y0(X0) ;
writeln('Invalid input. Please enter a number or "stop".'), tangent_line
) ).
read_y0(X0) :-
write('Enter the value of y0: '), read_line_to_codes(user_input, Y0_input),
atom_codes(Y0_atom, Y0_input),
(Y0_atom == stop ->
halt ;
(atom_number(Y0_atom, Y0) ->
read_slope(X0, Y0);
writeln('Invalid input. Please enter a number or "stop".'), read_y0(X0)
) ).
read_slope(X0, Y0) :-
write('Enter the slope (m) (or type "stop" to finish): '), read_line_to_codes(user_input,
atom_codes(M_atom, M_input),
(M_atom == stop ->
halt ;
(atom_number(M_atom, M) ->
read_x(X0, Y0, M) ;
writeln('Invalid input. Please enter a number or "stop".'), read_slope(X0, Y0)
) ).
read_x(X0, Y0, M) :-
write('Enter the value of x (or type "stop" to finish): '), read_line_to_codes(user_input
atom_codes(Atom, Input),
(Atom == stop ->
halt ;
(atom_number(Atom, X) ->
calculate_y(X0, Y0, M, X, Y),
format('The value of y at x = ~2f is: ~2f~n', [X, Y]),
read_x(X0, Y0, M) ;
writeln('Invalid input. Please enter a number or "stop".'), read_x(X0, Y0, M)
)).
calculate_y(X0, Y0, M, X, Y) :-
Y is Y0 + M * (X - X0).
main :-
tangent_line.
:- initialization(main, main).
```

```
Enter the value of x0: 2
Enter the value of y0: 3
Enter the slope (m) (or type "stop" to finish): 2
Enter the value of x (or type "stop" to finish): 4
The value of y at x = 4.00 is: 7.00
Enter the value of x (or type "stop" to finish): stop
```

9-Month Name

This code is a simple program that takes a user-inputted month number (1-12) and outputs the corresponding month name.

```
month_name_loop :-
writeln('Enter a month number : '),
read_month_number(MonthNumber),
( MonthNumber == stop ->
true
;
( month_name(MonthNumber, MonthName) ->
format('Month name: ~w~n', [MonthName])
;
writeln('Invalid month number')
),
month_name_loop
).
read_month_number(MonthNumber) :-
read_line_to_codes(user_input, Input),
atom_codes(Atom, Input),
( Atom == 'stop' ->
MonthNumber = stop
;
atom_number(Atom, MonthNumber)
).
month_name(1, 'January').
month_name(2, 'February').
month_name(3, 'March').
month_name(4, 'April').
month_name(5, 'May').
month_name(6, 'June').
month_name(7, 'July').
month_name(8, 'August').
month_name(9, 'September').
month_name(10, 'October').
month_name(11, 'November').
month_name(1, 'January').
month_name(2, 'February').
month_name(3, 'March').
month_name(4, 'April').
month_name(5, 'May').
month_name(6, 'June').
month_name(7, 'July').
month_name(8, 'August').
month_name(9, 'September').
month_name(10, 'October').
month_name(11, 'November').
month_name(12, 'December').
Example eighth :

Program to determine the month name from entered number:

| Page 38

main :-
month_name_loop,
halt.
:- initialization(main, main).
```

```
Enter a month number :
|: 8
Month name: August
Enter a month number :
|: stop


...Program finished with exit code 0
Press ENTER to exit console.
```

10- Calculating the factorial number

A program that calculates the product of all positive integers up to a given number. For example, the factorial of 5 is 5 * 4 * 3 * 2 * 1 = 120.

Program

```
1 fact(0, 1).
2
3 fact(N, Result) :-
4     N > 0,
5     N1 is N - 1,
6     fact(N1, R1),
7     Result is N * R1.
```

fact(5 , Result).

Result = 120

Next | 10 | 100 | 1,000 | Stop

```
?- fact(5 , Result).
```

11- Simple calculator

A simple calculator program allows the user to perform basic arithmetic operations (addition, subtraction, multiplication, and division) between two numbers.

Program

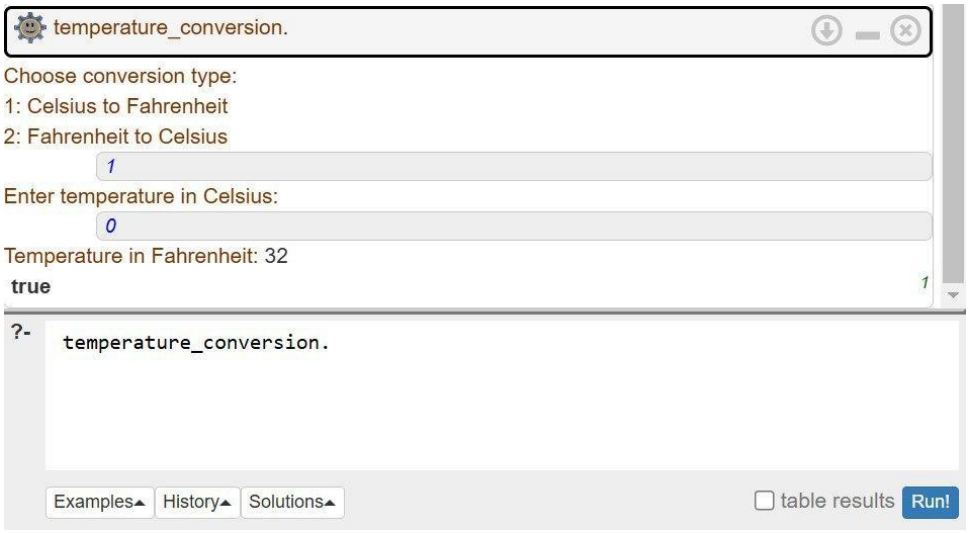
```
simpleCalc :-
    write('Enter the first number: '), nl,
    read(Num1),
    write('Enter the operator (+, -, *, /): '), nl,
    read(Op),
    write('Enter the second number: '), nl,
    read(Num2),
    ( Op = '+'
      -> Result is Num1 + Num2
    ; Op = '-'
      -> Result is Num1 - Num2
    ; Op = '*'
      -> Result is Num1 * Num2
    ; Op = '/'
      -> ( Num2 =\= 0
           -> Result is Num1 / Num2
           ; write('Error: Division by zero.'), nl, fail
         )
    ; write('Error: Invalid operator.'), nl, fail
    ),
    write('The result is: '), write(Result), nl.
```

12- Temperature conversion program

This program converts temperatures between Celsius and Fahrenheit. The user is prompted to select a conversion type: 1.Celsius to Fahrenheit 2.Fahrenheit to Celsius. Based on the choice, the program reads the input temperature and converts it using the appropriate formula: -Celsius to Fahrenheit: F = (C * 9/5) + 32 - Fahrenheit to Celsius: C = (F - 32) * 5/9 ,finally, the program prints the converted temperature.

Program

```
celToFah(Celsius, Fahrenheit) :-
    Fahrenheit is Celsius * 9 / 5 + 32.
fahToCel(Fahrenheit, Celsius) :-
    Celsius is (Fahrenheit - 32) * 5 / 9.
temperature_conversion :-
    write('Choose conversion type:'), nl,
    write('1: Celsius to Fahrenheit'), nl,
    write('2: Fahrenheit to Celsius'), nl,
    read(Choice),
    ( Choice =:= 1
      -> write('Enter temperature in Celsius: '), nl,
         read(Celsius),
         celToFah(Celsius, Fahrenheit),
         write('Temperature in Fahrenheit: '), write(Fahrenheit), nl
    ; Choice =:= 2
      -> write('Enter temperature in Fahrenheit: '), nl,
         read(Fahrenheit),
         fahToCel(Fahrenheit, Celsius),
         write('Temperature in Celsius: '), write(Celsius), nl ;
    write('Invalide choice'), nl, fail ).
```

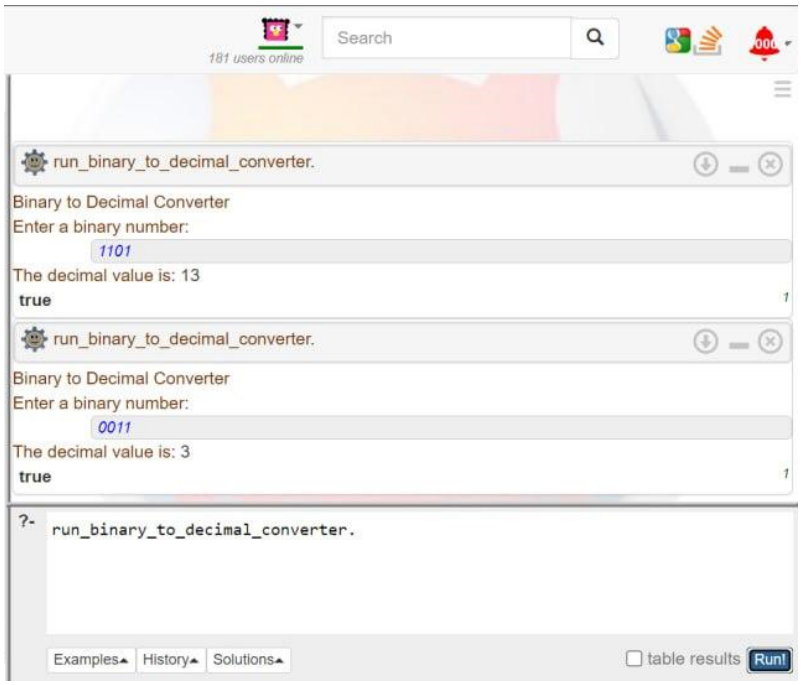


13- Simple password validator

This program allows the user to enter a password which then the program checks if the password is valid according to the conditions. Conditions for a valid password are: Should have at least one number. Should have at least one uppercase letter. Should be more or equal to 8 characters long.

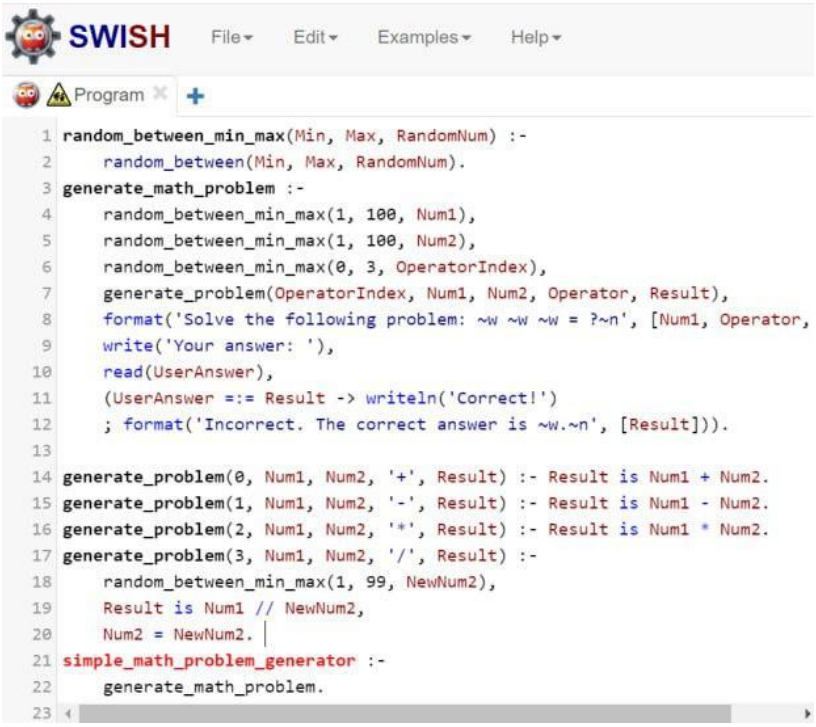
14- Binary to decimal converter

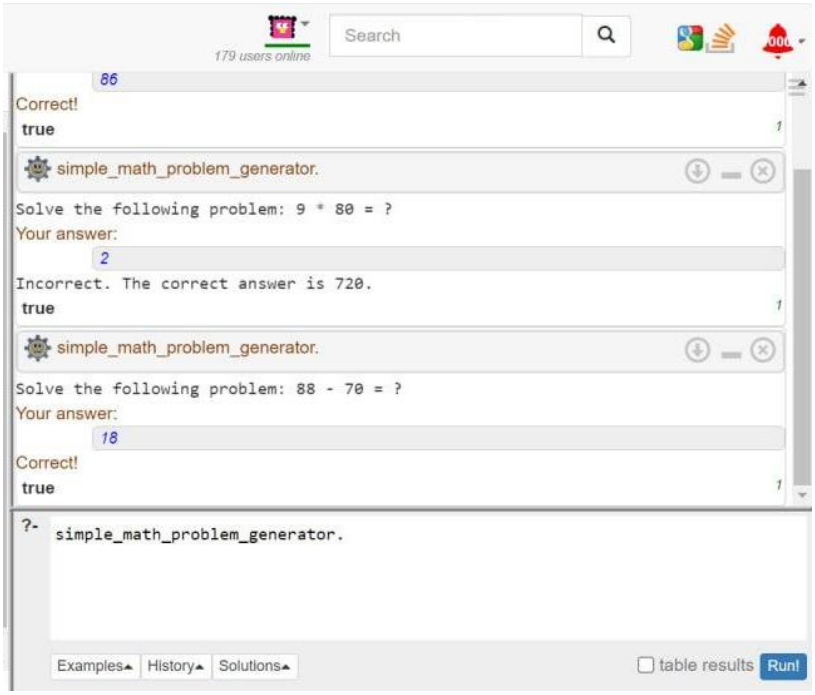
A program that converts a binary number into its decimal equivalent. It allows the user to enter a binary number, processes each bit from right to left, and calculates the corresponding decimal value by adding the powers of 2 for each '1' encountered. The final decimal result is then displayed to the user.



15- Simple math problem generator

A program that creates basic arithmetic problems by randomly selecting two integers (1 to 100) and one of four operations: addition, subtraction, multiplication, or division (with checks to avoid division by zero). Users solve the problem and receive instant feedback on their answer, including the correct solution if needed.

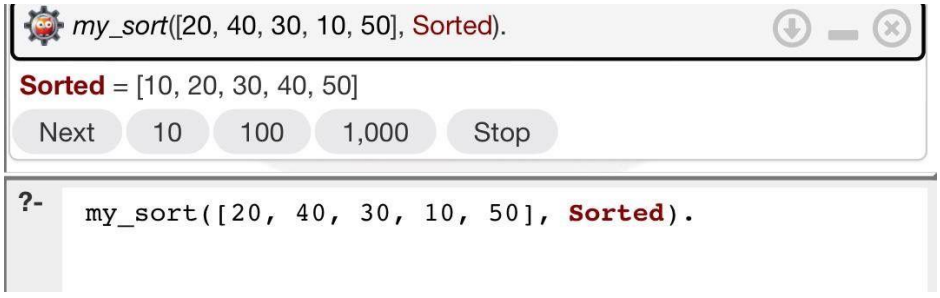




16- Sorting

This program implements the insertion sort algorithm to sort an array of integers. It defines an array of 5 elements, initially set to {10, 20, 30, 40, 50}. The sorting process moves through the array, shifting elements that are larger than the current key to the right, and placing the key in its correct position. The program prints the array before and after sorting.

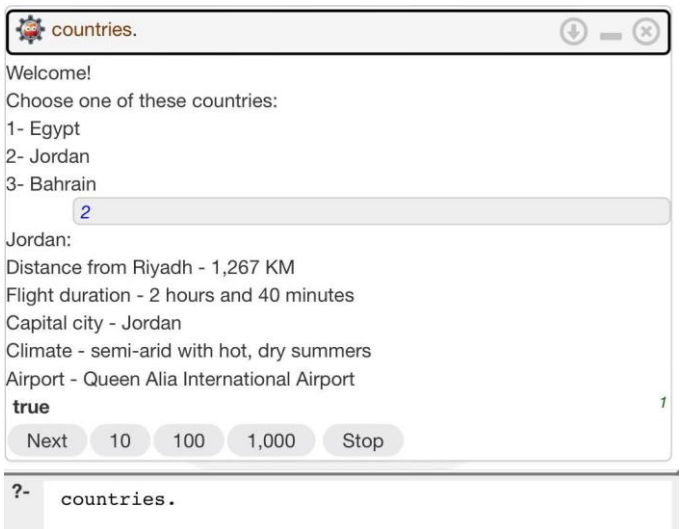
```
% find the minimum number in a list
min([N], N).
min([H1, H2 | T], N) :-
    H1 < H2,
    min([H1 | T], N).
min([H1, H2 | T], N) :-
    H1 > H2,
    min([H2 | T], N).

% removing the minimum number from the list
remove_element(X, [X | T], T).
remove_element(X, [H | T], [H | Res]) :-
    remove_element(X, T, Res).

% sorting the list
my_sort([], []).
my_sort(List, [X | Res]) :-
    min(List, X),
    remove_element(X, List, List2),
    my_sort(List2, Res).
```



17- Countries

This program prompts the user to select one of three countries (Egypt, Jordan, or Bahrain) and displays details about the chosen country, such as distance from Riyadh, flight duration, capital city, climate, and main airport. If the user enters an invalid number, the program shows an error message.

```
countries :-
    write("Welcome!"), nl,
    write("Choose one of these countries: "), nl,
    write("1- Egypt"), nl,
    write("2- Jordan"), nl,
    write("3- Bahrain"), nl,
    read(N),

    ( N =:= 1 ->
        write("Egypt:"), nl,
        write("Distance from Riyadh - 1,214 KM"), nl,
        write("Flight duration - 2 hours and 30 minutes"), nl,
        write("Capital city - Cairo"), nl,
        write("Climate - hot desert climate with mild winters"), nl,
        write("Airport - Cairo International Airport");
      N =:= 2 ->
        write("Jordan:"), nl,
        write("Distance from Riyadh - 1,267 KM"), nl,
        write("Flight duration - 2 hours and 40 minutes"), nl,
        write("Capital city - Amman"), nl,
        write("Climate - semi-arid with hot, dry summers"), nl,
        write("Airport - Queen Alia International Airport");
      N =:= 3 ->
        write("Bahrain:"), nl,
        write("Distance from Riyadh - 467 KM"), nl,
        write("Flight duration - 1 hour"), nl,
        write("Capital city - Manama"), nl,
        write("Climate - desert climate with very hot summers"), nl,
        write("Airport - Bahrain International Airport");
      write("Invalid choice, please enter 1, 2, or 3.")
    ).
```



18- Even and odd numbers

This program asks the user to enter six numbers, then displays them. It separates and prints the even and odd numbers based on a simple modulus check (mod 2). The program concludes with a thank-you message.

```
start :-
    write("Hello, please enter any 6 numbers"), nl,
    read(Value1),
    read(Value2),
    read(Value3),
    read(Value4),
    read(Value5),
    read(Value6),

    write("Original values are: "), nl,
    write(Value1), write(" "), write(Value2), write(" "),
    write(Value3), write(" "), write(Value4), write(" "),
    write(Value5), write(" "), write(Value6), nl,

    write("Even values are: "), nl,
    even_checker(Value1), write(" "), even_checker(Value2), write(" "),
    even_checker(Value3), write(" "), even_checker(Value4), write(" "),
    even_checker(Value5), write(" "), even_checker(Value6), nl,

    write("Odd values are: "), nl,
    odd_checker(Value1), write(" "), odd_checker(Value2), write(" "),
    odd_checker(Value3), write(" "), odd_checker(Value4), write(" "),
    odd_checker(Value5), write(" "), odd_checker(Value6), nl,

    write("Thank you!").

even_checker(X) :-
    X mod 2 =:= 0 -> write(X); write("").

odd_checker(X) :-
    X mod 2 =:= 1 -> write(X); write("").
```

```
start.
Hello, please enter any 6 numbers
4
5
6
9
4
2
Original values are:
4 5 6 9 4 2
Even values are:
4 6 4 2
Odd values are:
5 9
Thank you!
true 1

?- start.
```

Comparison (Pascal):

Readability:

High for structured and simple programs due to clear syntax, but verbose for complex applications.

Writability:

Easy for small, structured tasks but lacks flexibility for modern programming needs.

Reliability:

Strongly reliable due to strict syntax and strong typing, though less so for large, complex systems.

Cost:

Low-cost, with free compilers and a simple learning curve, but limited modern applicability may increase integration costs.

Portability:

Highly portable, with broad support across platforms, though minor compiler differences exist.

Generality:

Limited; best suited for education, algorithm development, and structured programming.

Well-Definedness:

Highly well-defined, with consistent syntax, though variations like Turbo Pascal introduce some differences.

Comparison (Scheme):

Readability:

Scheme’s syntax is uniform and simple, improving readability for those familiar with its structure, but challenging for beginners.

Writability:

Highly writable for functional programming tasks; expressive, concise, and flexible for symbolic manipulation and recursion-heavy problems.

Reliability:

Reliable when used in a functional paradigm, but lacks strict typing, which can lead to runtime errors.

Cost:

Low development cost for functional or symbolic tasks, moderate runtime cost due to typically slower performance.

Portability:

Portable in principle, but differences between implementations can limit portability for complex programs.

Generality:

Highly general-purpose, supporting multiple paradigms like functional, procedural, and symbolic programming.

Well-Definedness:

Clear and minimal specification, but inconsistencies can arise due to variations in implementations.

Comparison (Prolog):

Readability:

Prolog is readable within its domain, leveraging a clear declarative style, but it has a steep learning curve for beginners.

Writability:

Highly writable for rule-based, logic-driven applications but limited for procedural or state-intensive tasks.

Reliability:

Reliable for logic-based applications due to its declarative nature and absence of mutable state, though runtime errors related to unification or recursion are possible.

Cost:

Prolog has a low development cost for logic-heavy applications, moderate maintenance cost due to clear logical structure, but potentially high runtime costs due to computationally expensive backtracking.

Portability:

Moderately portable, but differences between implementations can limit consistency.

Generality:

Specialized for logic programming, AI, and knowledge representation; less suited for general-purpose tasks.

Well-Definedness:

Declarative syntax and semantics are clear, but implementation-specific features can cause inconsistencies.

Final Comparison:

Pascal is the easiest language to learn, read, and write, especially for beginners, due to its clear and structured syntax designed for procedural programming. Scheme is more powerful and concise but harder to grasp initially because of its functional paradigm and heavy use of parentheses. Prolog, while ideal for solving logic-based problems like AI, is the most challenging to learn due to its declarative approach and reliance on concepts like backtracking. In summary, Pascal is best for beginners, Scheme for functional programming enthusiasts, and Prolog for advanced logical problem-solving.

Conclusion:

In conclusion, understanding the procedural, functional, and logic programming types helps developers select the most suitable approach for their tasks. Procedural languages like Pascal are great for structured steps, functional languages like Scheme simplify complex computations, and logic languages like Prolog solve problems through reasoning. Choosing the right type ensures efficiency, low resource usage, and optimal solutions, allowing for versatile and maintainable software.