LEXICOM SAMPLE PROGRAMS

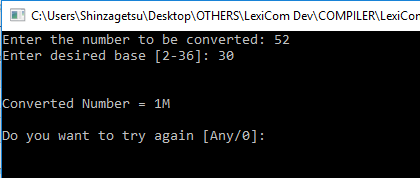
**PAMANTASAN NG LUNGSOD NG MAYNILA**

*(University of the City of Manila)*

Intramuros, Manila

1. Decimal to Any Base

Sample Input – Output:



Source Code:

Task Char Value(Int Index).

Lead:

Start

Array Char ConvNum of 36.

Var Int N2ConV; Base; Index is 0; NBase.

Var Char C.

Var String Resp.

Var Boolean Exit is No; Pass is No.

Do

Index = 0.

Clear.

Say "Enter the number to be converted: ".

Read N2ConV.

Do

Pass = No.

Say "Enter desired base [2-36]: ".

Read Base.

If(Base < 37)

If(Base > 1)

Pass = Yes.

Otherwise

Say "Base is too low! Try again...\n".

EndIf.

Otherwise

Say "Base is too high! Try again...\n".

EndIf.

LoopIf(Pass != Yes).

Until (N2ConV != 0)

NBase = N2ConV % Base.

ConvNum[Index] = Value(NBase).

N2ConV = N2ConV / Base.

Index = Index + 1.

Loop.

Index = Index - 1.

Say "\n\nConverted Number = ".

For( ; Index>=0; Index--)

C = ConvNum[Index].

Say C.

Loop.

Say "\n\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.

Task Char Value:

Start

Var Char Str is '-'.

Option Index

Start

State 0: Str = '0'. Stop. State 1: Str = '1'. Stop. State 2: Str = '2'. Stop. State 3: Str = '3'. Stop. State 4: Str = '4'. Stop. State 5: Str = '5'. Stop. State 6: Str = '6'. Stop. State 7: Str = '7'. Stop.

State 8: Str = '8'. Stop. State 9: Str = '9'. Stop. State 10: Str = 'A'. Stop. State 11: Str = 'B'. Stop. State 12: Str = 'C'. Stop. State 13: Str = 'D'. Stop. State 14: Str = 'E'. Stop. State 15: Str = 'F'. Stop.

State 16: Str = 'G'. Stop. State 17: Str = 'H'. Stop. State 18: Str = 'I'. Stop. State 19: Str = 'J'. Stop. State 20: Str = 'K'. Stop. State 21: Str = 'L'. Stop. State 22: Str = 'M'. Stop. State 23: Str = 'N'. Stop.

State 24: Str = 'O'. Stop. State 25: Str = 'P'. Stop. State 26: Str = 'Q'. Stop. State 27: Str = 'R'. Stop. State 28: Str = 'S'. Stop. State 29: Str = 'T'. Stop. State 30: Str = 'U'. Stop. State 31: Str = 'V'. Stop.

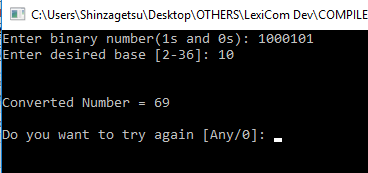
State 32: Str = 'W'. Stop. State 33: Str = 'X'. Stop. State 34: Str = 'Y'. Stop. State 35: Str = 'Z'. Stop.

End.

Response Str.

End.#

2. Binary to Any Base



Source Code:

Task Boolean ChkBin(Int Num).

Task Char Value(Int Index).

Task Int ToDec(Int Num).

Lead:

Start

Array Char ConvNum of 36.

Var Int N2ConV; Base; Index is 0; NBase.

Var Char C.

Var String Resp.

Var Boolean Exit is No; Pass is No; IsBin is No.

Do

Index = 0.

Clear.

Do

Say "Enter binary number(1s and 0s): ".

Read N2ConV.

IsBin = ChkBin(N2ConV).

If(IsBin == Yes)

N2ConV = ToDec(N2ConV).

Otherwise

Say "Input should be 1s and 0s only. Try again..\n".

EndIf.

LoopIf(IsBin != Yes).

Do

Pass = No.

Say "Enter desired base [2-36]: ".

Read Base.

If(Base < 37)

If(Base > 1)

Pass = Yes.

Otherwise

Say "Base is too low! Try again...\n".

EndIf.

Otherwise

Say "Base is too high! Try again...\n".

EndIf.

LoopIf(Pass != Yes)

Until (N2ConV != 0)

NBase = N2ConV % Base.

ConvNum[Index] = Value(NBase).

N2ConV = N2ConV / Base.

Index = Index + 1.

Loop.

Index = Index - 1.

Say "\n\nConverted Number = ".

For( ; Index>=0; Index--)

C = ConvNum[Index].

Say C.

Loop.

Say "\n\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.

Task Int ToDec:

Start

Var Int Bin; Dec is 0; Rem; Base is 1.

Bin = Num.

Until (Num > 0)

Rem = Num % 10.

Dec = Dec + Rem \* Base.

Base = Base \* 2.

Num = Num / 10.

Loop.

Response Dec.

End.

Task Boolean ChkBin:

Start

Var Boolean Pass is Yes.

Var Int Dv.

Until(Num != 0)

Dv = Num % 10.

If(Dv>1)

Pass = No.

Stop.

EndIf.

Num = Num / 10.

Loop.

Response Pass.

End.

Task Char Value:

Start

Var Char Str is '-'.

Option Index

Start

State 0: Str = '0'. Stop. State 1: Str = '1'. Stop. State 2: Str = '2'. Stop. State 3: Str = '3'. Stop. State 4: Str = '4'. Stop. State 5: Str = '5'. Stop. State 6: Str = '6'. Stop. State 7: Str = '7'. Stop.

State 8: Str = '8'. Stop. State 9: Str = '9'. Stop. State 10: Str = 'A'. Stop. State 11: Str = 'B'. Stop. State 12: Str = 'C'. Stop. State 13: Str = 'D'. Stop. State 14: Str = 'E'. Stop. State 15: Str = 'F'. Stop.

State 16: Str = 'G'. Stop. State 17: Str = 'H'. Stop. State 18: Str = 'I'. Stop. State 19: Str = 'J'. Stop. State 20: Str = 'K'. Stop. State 21: Str = 'L'. Stop. State 22: Str = 'M'. Stop. State 23: Str = 'N'. Stop.

State 24: Str = 'O'. Stop. State 25: Str = 'P'. Stop. State 26: Str = 'Q'. Stop. State 27: Str = 'R'. Stop. State 28: Str = 'S'. Stop. State 29: Str = 'T'. Stop. State 30: Str = 'U'. Stop. State 31: Str = 'V'. Stop.

State 32: Str = 'W'. Stop. State 33: Str = 'X'. Stop. State 34: Str = 'Y'. Stop. State 35: Str = 'Z'. Stop.

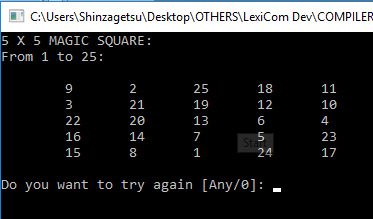
End.

Response Str.

End.#

3. Magic Square

Sample Input – Output:



Source Code:

Task Null Init(Int Size; Int S).

Lead:

Start

Var Int S; Size is 0; Odd.

Var Boolean Pass is No.

Var String Resp.

Var Boolean Exit is No.

Do

Clear.

Do

Clear.

Say "Enter the size of the magic square [3-15]: \n",

"NOTE: SIZE SHOULD BE ODD NUMBERS ONLY.\n".

Read Size.

Odd = Size%2.

If ((Size<3)||(Size>15))

Say "ERROR: SIZE IS OUT OF RANGE!!! Press any key to try again.". Read Resp.

Or (Odd==0)

Say "ERROR: SIZE SHOULD BE ODD NUMBER!!! Press any key to try again.". Read Resp.

Otherwise

Pass = Yes.

EndIf.

LoopIf(Pass != Yes).

Pass = No.

Do

Say "Enter the number to start: ".

Read S.

If(S != 0)

Pass = Yes.

Otherwise

Say "Starting number cannot be zero. Try again..\n".

EndIf.

LoopIf(Pass != Yes).

Init(Size;S).

Say "\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.

Task Null Init:

Start

Array Int MS of 15 by 15.

Var Int Lim; Val1; Val2; I; J; Num.

Lim = S + Size\*Size.

Var Int Index1; Index2.

Var Boolean Check is No.

Val1 = Size - 1.

Do

Say "Enter starting position 1 (0-", Val1, "): ".

Read Index1.

If(Index1 < Size)

If(Index1 > ~1)

Check = Yes.

Otherwise

Say "Starting Position Out of Range! Try again..\n".

EndIf.

Otherwise

Say "Starting Position Out of Range! Try again..\n".

EndIf.

LoopIf(Check != Yes).

Check = No.

Do

Say "Enter starting position 2 (0-", Val1, "): ".

Read Index2.

If(Index2 < Size)

If(Index2 > ~1)

Check = Yes.

Otherwise

Say "Starting Position Out of Range! Try again..\n".

EndIf.

Otherwise

Say "Starting Position Out of Range! Try again..\n".

EndIf.

LoopIf(Check != Yes).

Clear.

Say Size, " X ", Size, " MAGIC SQUARE:\n".

Val2 = Lim - 1.

Say "From ", S, " to ", Val2, ":\n\n ".

For(I = 0; I < Size; I++)

For(J = 0; J < Size; J++)

MS[I][J] = 0.

Loop.

Loop.

PS J = Size/2. I = Size-1.

I = Index1. J = Index2.

For(Num = S; Num < Lim; Num++)

MS[I][J] = Num.

I = I + 1.

If(I == Size) I=0. EndIf.

J = J - 1.

If(J < 0) J=Size-1. EndIf.

Val1 = MS[I][J].

If(Val1 != 0)

I = I - 2.

J = J + 1.

If(I < 0) I = I + Size. EndIf.

If(J >= Size) J = J - Size. EndIf.

If(Num != Val2)

MS[I][J] = Num.

EndIf.

EndIf.

Loop.

PS J = Size/2. MS[0][J] = Num. Num = 1.

For(I = 0; I < Size; I++)

For(J = 0; J < Size; J++)

Val2 = MS[I][J].

Say "\t", Val2.

Loop.

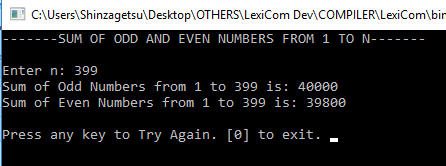
Say "\n ".

Loop.

End.

4. 4. Sum of Odd Even Numbers from 1 to N

Sample Input – Output:



Source Code:

Lead:

Start

Var Int N; I; Sum1 is 0; Sum2 is 0.

Var String X is " ".

Var Boolean Resp is Yes.

Do

Say "-------SUM OF ODD AND EVEN NUMBERS FROM 1 TO N-------\n\n".

Say "Enter n: ".

Read N.

For(I=1; I<=N; I++)

Sum1 = Sum1 + I.

I =I+1.

Loop.

For(I=2; I<=N; I++)

Sum2 = Sum2 + I.

I =I+1.

Loop.

Say "Sum of Odd Numbers from 1 to ", N, " is: ", Sum1.

Say "\nSum of Even Numbers from 1 to ", N, " is: ", Sum2.

Say "\n\nPress any key to Try Again. [0] to exit. ".

Read X.

If(X == "0")

Resp = No.

Otherwise

Resp = Yes.

Clear.

Sum1 = 0.

Sum2 = 0.

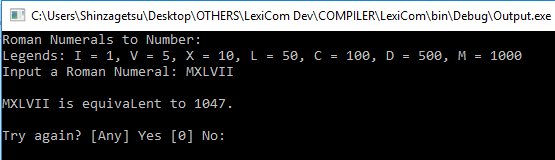
EndIf.

LoopIf (Resp == Yes).

End.#

5. Roman Numeral to Number

Sample Input – Output:



Source Code:

Task Null Rom2Dec(String Msg).

Task Boolean ChkInv(String RomStr).

Task Boolean ChkRep(String RomStr).

Task Int Translate(String RomStr).

Lead:

Start

Var String Msg is "Roman Numerals to Number:\nLegends: I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000\nInput a Roman Numeral: ".

Rom2Dec(Msg).

End.

Task Null Rom2Dec:

Start

Var Int Dec.

Array Char RomChr of 100.

Var String RomStr; Resp.

Var Boolean IsInv; IsRep; Exit is No.

Do

Dec = ~1.

Clear.

Say Msg.

Read RomStr.

IsInv = ChkInv(RomStr).

If (IsInv == No)

IsRep = ChkRep(RomStr).

If (IsRep == No)

Dec = Translate(RomStr).

Otherwise

Say "\nInvalid input! A numeral was used more than the allowed times..".

EndIf.

Otherwise

Say "\nInvalid input! Invalid character found..".

EndIf.

If (Dec == ~1)

Say "\nInvalid input! Invalid numeral sequence..".

Otherwise

Say "\n", RomStr, " is equivaLent to ", Dec, ".".

EndIf.

Say "\n\nTry again? [Any] Yes [0] No: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

LoopIf (Exit != Yes).

Say "Program will terminate at enter..".

End.

Task Boolean ChkInv:

Start

Var Int I; Len.

Array Char RomChr of 100.

Var Char T.

Var Boolean IsInv is No.

RomChr = ToCharArray(RomStr).

Len = StrLen(RomStr).

For (I = 0; I < Len; I++)

T = RomChr[I].

If ((T != 'i' ) && ( T != 'I' ) && ( T != 'v' ) && ( T != 'V' ) && ( T != 'x' ) && ( T != 'X' ) && ( T != 'l' ) && ( T != 'L' ) && ( T != 'c' ) && ( T != 'C' ) && ( T != 'd' ) && ( T != 'D' ) && ( T != 'm' ) && ( T != 'M'))

IsInv = Yes.

Stop.

EndIf.

Loop.

Response IsInv.

End.

Task Boolean ChkRep:

Start

Var Int I; Len; CTI is 0; CTV is 0; CTX is 0; CTL is 0; CTC is 0; CTD is 0; CTM is 0.

Array Char RomChr of 100.

Var Char Prev is ' '; ValC.

Var Boolean IsRep is No.

RomChr = ToCharArray(RomStr).

Len = StrLen(RomStr).

For (I = 0; I < Len; I++)

ValC = RomChr[I].

If (Prev == ValC)

Option ValC

Start

State 'm': CTM = CTM+1. Stop.

State 'M': CTM = CTM+1. Stop.

State 'd': CTD = CTD+1. Stop.

State 'D': CTD = CTD+1. Stop.

State 'c': CTC = CTC+1. Stop.

State 'C': CTC = CTC+1. Stop.

State 'l': CTL = CTL+1. Stop.

State 'L': CTL = CTL+1. Stop.

State 'x': CTX = CTX+1. Stop.

State 'X': CTX = CTX+1. Stop.

State 'v': CTV = CTV+1. Stop.

State 'V': CTV = CTV+1. Stop.

State 'i': CTI = CTI+1. Stop.

State 'I': CTI = CTI+1. Stop.

End.

Otherwise

CTI = 0. CTV = 0. CTX = 0. CTL = 0.

CTC = 0. CTD = 0. CTM = 0.

EndIf.

Prev = RomChr[I].

If ((CTI >= 3) || (CTV >= 1) || (CTX >= 3) || (CTL >= 1) || (CTC >= 3) || (CTD >= 1) || (CTM >= 3))

IsRep = Yes. Stop.

EndIf.

Loop.

Response IsRep.

End.

Task Int Translate:

Start

Var Int I; Len; Dec is 0.

Array Char RomChr of 100.

Var Char P is ' '; T.

RomChr = ToCharArray(RomStr).

Len = StrLen(RomStr).

Len = Len - 1.

For (I = Len; I >= 0; I--)

T = RomChr[I].

If ((T == 'i') || (T == 'I'))

If ((P == 'i') || (P == 'I') || (P == ' '))

Dec = Dec + 1.

Or ((P == 'x') || (P == 'X') || (P == 'v') || (P == 'V'))

Dec = Dec - 1.

Otherwise

Dec = ~1. Stop.

EndIf.

Or ((T == 'v') || (T == 'V'))

If ((P == 'x') || (P == 'X') || (P == 'l') || (P == 'L') || (P == 'c') || (P == 'C') || (P == 'd') || (P == 'D') || (P == 'm') || (P == 'M'))

Dec = ~1. Stop.

Otherwise

Dec = Dec + 5.

EndIf.

Or ((T == 'x') || (T == 'X'))

If ((P == 'c') || (P == 'C') || (P == 'l') || (P == 'L'))

Dec = Dec - 10.

Or ((P == 'x') || (P == 'X') || (P == 'i') || (P == 'I') || (P == 'v') || (P == 'V') || (P == ' '))

Dec = Dec + 10.

Otherwise

Dec = ~1. Stop.

EndIf.

Or ((T == 'l') || (T == 'L'))

If ((P == 'c') || (P == 'C') || (P == 'd') || (P == 'D') || (P == 'm') || (P == 'M'))

Dec = ~1. Stop.

Otherwise

Dec = Dec + 50.

EndIf.

Or ((T == 'c') || (T == 'C'))

If ((P == 'm') || (P == 'M') || (P == 'd') || (P == 'D'))

Dec = Dec - 100.

Or ((P == 'c') || (P == 'C') || (P == ' '))

Dec = Dec + 100.

Otherwise

Dec = ~1. Stop.

EndIf.

Or ((T == 'd') || (T == 'D'))

If ((P == 'm') || (P == 'M'))

Dec = ~1. Stop.

Otherwise

Dec = Dec + 500.

EndIf.

Or ((T == 'm') || (T == 'M'))

Dec = Dec + 1000.

EndIf.

P = RomChr[I].

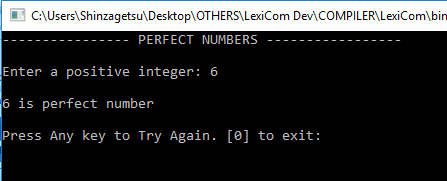
Loop.

Response Dec.

End.#

6. Perfect Number

Sample Input – Output:



Source Code:

Task Int CheckInput\_I(String Msg; String MsgMin; String MsgMax; Int Min; Int Max).

Lead:

Start

Var Int Number; Number2; Rem; Sum is 0; I.

Var String Resp.

Var Boolean Exit is No.

Do

Number = 0.

Rem = 0.

Sum = 0.

Clear.

Say "---------------- PERFECT NUMBERS -----------------".

Number = CheckInput\_I("\n\nEnter a positive integer: "; "\n\nNo negative input allowed. Please Try Again...\n"; "\n\nInput cannot exceed 100,000,000. Please Try Again...\n"; 0; 100000000).

Number2 = Number - 1.

Say "".

For (I = 1; I <= Number2 ; I++)

Rem = Number % I.

If (Rem == 0)

Sum = Sum + I.

EndIf.

Loop.

If (Sum == Number)

Say "\n", Number, " is perfect number".

Otherwise

Say "\n", Number, " is not a perfect number".

EndIf.

Say "\n\nPress Any key to Try Again. [0] to exit: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

LoopIf ( Exit == No ).

Say "\n\nProgram will terminate in key press...".

End.

Task Int CheckInput\_I:

Start

Var Int Ins.

Var Boolean Pass is Yes.

Do

Say Msg.

Read Ins.

If(Ins > Max)

Say MsgMax.

Pass = No.

Or(Ins < Min)

Say MsgMin.

Pass = No.

Otherwise

Pass = Yes.

EndIf.

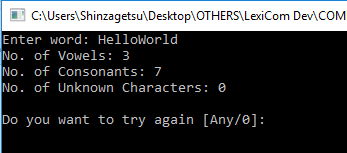
LoopIf(Pass == No).

Response Ins.

End.#

7. Consonants and Vowels

Sample Input – Output:



Source Code:

Task Boolean Cons(String S).

Task Boolean Vowel(String S).

Lead:

Start

Var String Str is ""; S; Resp.

Var Boolean Exit is No; IsVow is No; IsCons is No.

Var Int Len is 0; I; V is 0; C is 0; U is 0.

Do

Clear.

Say "Enter word: ".

Read Str.

V = 0. C = 0. U = 0.

Len = StrLen(Str).

If(Len != 0)

For(I = 0; I < Len; I++)

S = Substr(Str; I; 1).

IsVow = Vowel(S).

If(IsVow == No)

IsCons = Cons(S).

If(IsCons == No)

U = U + 1.

Otherwise

C = C + 1.

EndIf.

Otherwise

V = V + 1.

EndIf.

Loop.

Say "No. of Vowels: ", V,

"\nNo. of Consonants: ", C,

"\nNo. of Unknown Characters: ", U.

Otherwise

Say "Please input a valid word...". Read Resp.

EndIf.

Say "\n\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.

Task Boolean Cons:

Start

Var Boolean Pass is No.

Option S

Start

State "B": Pass = Yes. Stop.

State "C": Pass = Yes. Stop.

State "D": Pass = Yes. Stop.

State "F": Pass = Yes. Stop.

State "G": Pass = Yes. Stop.

State "H": Pass = Yes. Stop.

State "J": Pass = Yes. Stop.

State "K": Pass = Yes. Stop.

State "L": Pass = Yes. Stop.

State "M": Pass = Yes. Stop.

State "N": Pass = Yes. Stop.

State "P": Pass = Yes. Stop.

State "Q": Pass = Yes. Stop.

State "R": Pass = Yes. Stop.

State "S": Pass = Yes. Stop.

State "T": Pass = Yes. Stop.

State "V": Pass = Yes. Stop.

State "W": Pass = Yes. Stop.

State "X": Pass = Yes. Stop.

State "Y": Pass = Yes. Stop.

State "Z": Pass = Yes. Stop.

State "b": Pass = Yes. Stop.

State "c": Pass = Yes. Stop.

State "d": Pass = Yes. Stop.

State "f": Pass = Yes. Stop.

State "g": Pass = Yes. Stop.

State "h": Pass = Yes. Stop.

State "j": Pass = Yes. Stop.

State "k": Pass = Yes. Stop.

State "l": Pass = Yes. Stop.

State "m": Pass = Yes. Stop.

State "n": Pass = Yes. Stop.

State "p": Pass = Yes. Stop.

State "q": Pass = Yes. Stop.

State "r": Pass = Yes. Stop.

State "s": Pass = Yes. Stop.

State "t": Pass = Yes. Stop.

State "v": Pass = Yes. Stop.

State "w": Pass = Yes. Stop.

State "x": Pass = Yes. Stop.

State "y": Pass = Yes. Stop.

State "z": Pass = Yes. Stop.

End.

Response Pass.

End.

Task Boolean Vowel:

Start

Var Boolean Pass is No.

Option S

Start

State "A": Pass = Yes. Stop.

State "E": Pass = Yes. Stop.

State "I": Pass = Yes. Stop.

State "O": Pass = Yes. Stop.

State "U": Pass = Yes. Stop.

State "a": Pass = Yes. Stop.

State "e": Pass = Yes. Stop.

State "i": Pass = Yes. Stop.

State "o": Pass = Yes. Stop.

State "u": Pass = Yes. Stop.

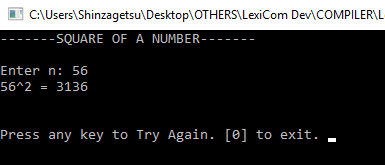
End.

Response Pass.

End.

8. Square of a Number

Sample Input – Output:



Source Code:

Lead:

Start

Var Int Num; Sqr.

Var String X is " ".

Var Boolean Resp is Yes.

Do

Say "-------SQUARE OF A NUMBER-------\n\n".

Say "Enter n: ".

Read Num.

Sqr = Num\*Num.

Say Num , "^2 = ", Sqr , "\n".

Say "\n\nPress any key to Try Again. [0] to exit. ".

Read X.

If(X == "0")

Resp = No.

Otherwise

Resp = Yes.

Clear.

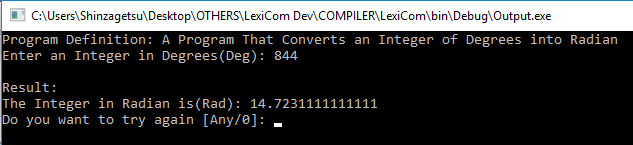
EndIf.

LoopIf (Resp == Yes).

End.#

9. Degree to Radian

Sample Input – Output:

.

Source Code:

Lead:

Start

Var Int Deg.

Var Double Rad.

Var Double Pi is 3.14.

Var String Resp.

Var Boolean Exit is No.

Do

Clear.

Say "Program Definition: A Program That Converts an Integer of Degrees into Radian\n".

Say "Enter an Integer in Degrees(Deg): ".

Read Deg.

Rad = Deg \* (Pi/180).

Say "\nResult:\n".

Say "The Integer in Radian is(Rad): ", Rad.

Say "\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

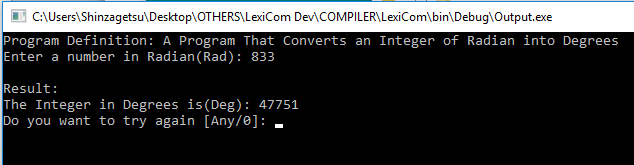
LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.#

10. Radian to Degrees

Sample Input – Output:



Source Code:

Lead:

Start

Var Double Deg.

Var Double Rad.

Var Double Pi is 3.14.

Var Double Mod is 0.0.

Var String Resp.

Var Boolean Exit is No.

Do

Clear.

Say "Program Definition: A Program That Converts an Integer of Radian into Degrees\n".

Say "Enter a number in Radian(Rad): ".

Read Rad.

Deg = Rad \* (180/Pi).

Mod = Deg%1.

Deg = Deg - Mod.

Say "\nResult:\n".

Say "The Integer in Degrees is(Deg): ", Deg.

Say "\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

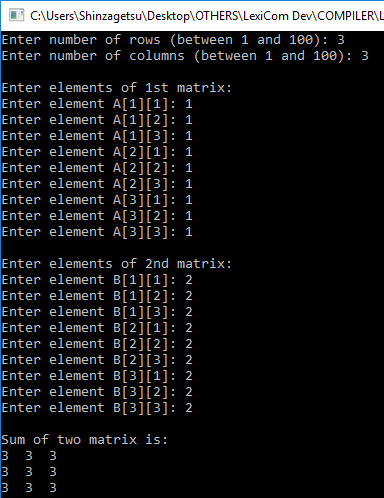
LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.#

11. Sum of two Matrix

Sample Input – Output:



Source Code:

Lead:

Start

Var Int R; C; I; J; Val1; Val2.

Array Int A of 100 by 100.

Array Int B of 100 by 100.

Array Int Sum of 100 by 100.

Say "Enter number of rows (between 1 and 100): ".

Read R.

Say "Enter number of columns (between 1 and 100): ".

Read C.

Say "\nEnter elements of 1st matrix: \n".

PS Storing elements of first matrix entered by user.

For(I = 0; I < R; ++I)

For(J = 0; J < C; ++J)

Val1 = I + 1. Val2 = J + 1.

Say "Enter element A[", Val1, "][", Val2, "]: ".

Read Val1.

A[I][J] = Val1.

Loop.

Loop.

PS Storing elements of second matrix entered by user.

Say "\nEnter elements of 2nd matrix: \n".

For(I = 0; I < R; ++I)

For(J = 0; J < C; ++J)

Val1 = I + 1. Val2 = J + 1.

Say "Enter element B[", Val1, "][", Val2, "]: ".

Read Val1.

B[I][J] = Val1.

Loop.

Loop.

PS Adding Two matrices

For(I = 0; I < R; ++I)

For(J = 0; J < C; ++J)

Sum[I][J] = A[I][J] + B[I][J].

Loop.

Loop.

PS Displaying the resultant sum matrix.

Say "\nSum of two matrix is: \n".

For(I = 0; I < R; ++I)

For(J = 0; J < C; ++J)

Val1 = Sum[I][J].

Say Val1, " ".

Val2 = C - 1.

If(J == Val2)

Say "\n".

EndIf.

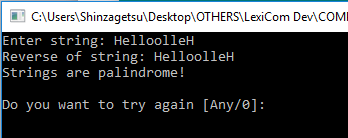
Loop.

Loop.

End.#

12. Palindrome

Sample Input – Output:



Source Code:

Lead:

Start

Var String Str is ""; Out is ""; Rvr is ""; Resp.

Var Boolean Exit is No.

Var Int Len is 0; I.

Do

Clear.

Rvr = "".

Say "Enter string: ".

Read Str.

Len = StrLen(Str) - 1.

If(Len >= 0)

Say "Reverse of string: ".

For(I = Len; I>=0; I--)

Out = Substr(Str; I; 1).

Rvr = Rvr + Out.

Say Out.

Loop.

If(Rvr == Str)

Say "\nStrings are palindrome!".

Otherwise

Say "\nStrings are not palindrome!".

EndIf.

Otherwise

Say "String cannot be reversed.".

EndIf.

Say "\n\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

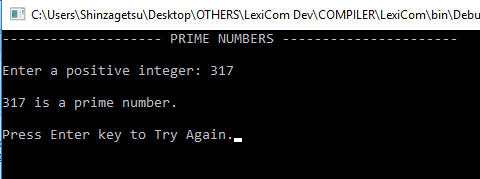
LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.#

13. Prime Numbers

Sample Input – Output:



Source Code:

Task Int CheckInput\_I(String Msg; String MsgMin; String MsgMax; Int Min; Int Max).

Task Null PrimeNumbers(Int N).

Lead:

Start

Var Int N.

Var Boolean Pass is Yes.

Var String Resp; CH.

Do

Clear.

Say "-------------------- PRIME NUMBERS ----------------------",

"\n\n[1] To Start",

"\n[0] Exit",

"\n\nEnter you choice: ".

Read CH.

Option CH

Start

State "1":

Clear.

Say "-------------------- PRIME NUMBERS ----------------------\n".

N = CheckInput\_I("\nEnter a positive integer: "; "\nInput must be a positive integer. Please Try Again...\n"; "Input cannot exceed 100,000,000. Please Try Again...\n"; 0; 100000000).

PrimeNumbers(N).

Say "\n\nPress Enter key to Try Again.".

Read Resp.

Stop.

State "0":

Pass = No.

Stop.

Default:

Say "Invalid Input! Press Enter to Try Again!".

Read Resp.

Pass = Yes.

End.

LoopIf ( Pass == Yes ).

Say "\nProgram will terminate in key press ...".

End.

Task Null PrimeNumbers:

Start

Var Int N2; I; Result.

Var Boolean IsPrime is No.

N2 = N/2.

If ( ( N == 1) || ( N == 0 ) )

Say "\n", N, " is not a prime number.".

Otherwise

For ( I=2; I<=N2; ++I)

Result = N % I.

If ( Result == 0 )

IsPrime = Yes.

EndIf.

Loop.

If ( IsPrime == Yes )

Say "\n", N, " is not a prime number.".

Otherwise

Say "\n", N, " is a prime number.".

EndIf.

EndIf.

End.

Task Int CheckInput\_I:

Start

Var Int Ins.

Var Boolean Pass is Yes.

Do

Say Msg.

Read Ins.

If(Ins > Max)

Say MsgMax.

Pass = No.

Or(Ins < Min)

Say MsgMin.

Pass = No.

Otherwise

Pass = Yes.

EndIf.

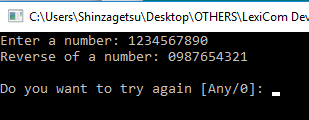
LoopIf(Pass == No).

Response Ins.

End.#

14. Reverse of a Number

Sample Input – Output:



Source Code:

Lead:

Start

Var Int Num is 0; Rvr is 0; Temp is 0.

Var String Resp.

Var Boolean Neg; Exit is No.

Var Int Len is 0; I.

Do

Clear.

Neg = No. Rvr = 0.

Say "Enter a number: ".

Read Num.

Say "Reverse of a number: ".

If(Num < 0)

Neg = Yes.

Num = Num \* ~1.

EndIf.

Until(Num != 0)

Temp = Num % 10.

Say Temp.

PS Rvr = (Rvr \* 10) + Temp .

Num = Num / 10.

Loop.

PS Say Rvr.

If(Neg == Yes)

Say "-".

EndIf.

Say "\n\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

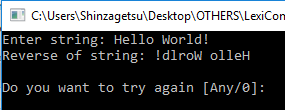
LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.#

15. Reverse of a String

Sample Input – Output:



Source Code:

Lead:

Start

Var String Str is ""; Out is ""; Rvr is ""; Resp.

Var Boolean Exit is No.

Var Int Len is 0; I.

Do

Clear.

Rvr = "".

Say "Enter string: ".

Read Str.

Len = StrLen(Str) - 1.

If(Len >= 0)

Say "Reverse of string: ".

For(I = Len; I>=0; I--)

Out = Substr(Str; I; 1).

Rvr = Rvr + Out.

Say Out.

Loop.

Otherwise

Say "String cannot be reversed.".

EndIf.

Say "\n\nDo you want to try again [Any/0]: ".

Read Resp.

If(Resp == "0")

Exit = Yes.

EndIf.

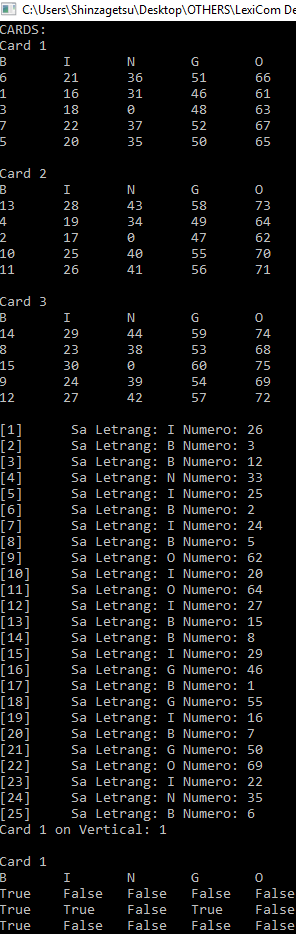
LoopIf(Exit != Yes).

Say "Program will terminate at enter...".

End.#

16. If Triangle can be Formed

Sample Input – Output:



Source Code:

Task String Letter(Int N).

Task Boolean P1(Boolean A;Boolean B;Boolean C;Boolean D;Boolean E).

Task Boolean P2(Boolean A;Boolean B;Boolean C;Boolean D).

Lead:

Start

Array Int C1 of 5 by 5.

Array Int C2 of 5 by 5.

Array Int C3 of 5 by 5.

Array Boolean CB1 of 5 by 5.

Array Boolean CB2 of 5 by 5.

Array Boolean CB3 of 5 by 5.

Array Int B of 15.

Array Int I of 15.

Array Int N of 15.

Array Int G of 15.

Array Int O of 15.

Array Int Set of 75.

Set = Random(1;75).

B = Random(1;15).

I = Random(16;30).

N = Random(31;45).

G = Random(46;60).

O = Random(61;75).

Var Int X; Y.

For(X=0;X<5;X++)

C1[0][X] = B[X]. C2[0][X] = B[X+5]. C3[0][X] = B[X+10].

C1[1][X] = I[X]. C2[1][X] = I[X+5]. C3[1][X] = I[X+10].

C1[2][X] = N[X]. C2[2][X] = N[X+5]. C3[2][X] = N[X+10].

C1[3][X] = G[X]. C2[3][X] = G[X+5]. C3[3][X] = G[X+10].

C1[4][X] = O[X]. C2[4][X] = O[X+5]. C3[4][X] = O[X+10].

For(Y=0;Y<5;Y++)

CB1[X][Y] = No.

CB2[X][Y] = No.

CB3[X][Y] = No.

Loop.

Loop.

Var Int Ctr is 0; NUM; DISP; Ctr1.

Var String LETTER; BingoCard is "".

Var Boolean Bingo is No; VAR\_A; VAR\_B; VAR\_C; VAR\_D; VAR\_E.

C1[2][2] = 0.

C2[2][2] = 0.

C3[2][2] = 0.

Say "CARDS:\n".

Say "Card 1\nB\tI\tN\tG\tO\n".

For(X = 0; X < 5; X++)

For(Y = 0; Y < 5; Y++)

Say C1[Y][X] , "\t".

Loop.

Say "\n".

Loop.

Say "\nCard 2\nB\tI\tN\tG\tO\n".

For(X = 0; X < 5; X++)

For(Y = 0; Y < 5; Y++)

Say C2[Y][X] , "\t".

Loop.

Say "\n".

Loop.

Say "\nCard 3\nB\tI\tN\tG\tO\n".

For(X = 0; X < 5; X++)

For(Y = 0; Y < 5; Y++)

Say C3[Y][X] , "\t".

Loop.

Say "\n".

Loop.

Say "\n".

Do

NUM = Set[Ctr].

DISP = Ctr + 1.

LETTER = Letter(NUM).

Say "[",DISP,"]\t Sa Letrang: ", LETTER, " Numero: ", NUM, "\n".

Option LETTER

Start

State "B":

For(X=0;X<5;X++)

If(C1[0][X] == NUM) CB1[0][X] = Yes. EndIf.

If(C2[0][X] == NUM) CB2[0][X] = Yes. EndIf.

If(C3[0][X] == NUM) CB3[0][X] = Yes. EndIf.

Loop.

Stop.

State "I":

For(X=0;X<5;X++)

If(C1[1][X] == NUM) CB1[1][X] = Yes. EndIf.

If(C2[1][X] == NUM) CB2[1][X] = Yes. EndIf.

If(C3[1][X] == NUM) CB3[1][X] = Yes. EndIf.

Loop.

Stop.

State "N":

For(X=0;X<5;X++)

If(C1[2][X] == NUM) CB1[2][X] = Yes. EndIf.

If(C2[2][X] == NUM) CB2[2][X] = Yes. EndIf.

If(C3[2][X] == NUM) CB3[2][X] = Yes. EndIf.

Loop.

Stop.

State "G":

For(X=0;X<5;X++)

If(C1[3][X] == NUM) CB1[3][X] = Yes. EndIf.

If(C2[3][X] == NUM) CB2[3][X] = Yes. EndIf.

If(C3[3][X] == NUM) CB3[3][X] = Yes. EndIf.

Loop.

Stop.

State "O":

For(X=0;X<5;X++)

If(C1[4][X] == NUM) CB1[4][X] = Yes. EndIf.

If(C2[4][X] == NUM) CB2[4][X] = Yes. EndIf.

If(C3[4][X] == NUM) CB3[4][X] = Yes. EndIf.

Loop.

Stop.

End.

For(Ctr1 = 0; Ctr1 < 5; Ctr1++)

VAR\_A = CB1[Ctr1][0].

VAR\_B = CB1[Ctr1][1].

VAR\_C = CB1[Ctr1][2].

VAR\_D = CB1[Ctr1][3].

VAR\_E = CB1[Ctr1][4].

Bingo = P1(VAR\_A;VAR\_B;VAR\_C;VAR\_D;VAR\_E).

If(Bingo == Yes)

Ctr1 = Ctr1 + 1.

BingoCard = "Card 1 on Vertical: ".

BingoCard = BingoCard + Ctr1. Stop.

EndIf.

VAR\_A = CB2[Ctr1][0].

VAR\_B = CB2[Ctr1][1].

VAR\_C = CB2[Ctr1][2].

VAR\_D = CB2[Ctr1][3].

VAR\_E = CB2[Ctr1][4].

Bingo = P1(VAR\_A;VAR\_B;VAR\_C;VAR\_D;VAR\_E).

If(Bingo == Yes)

Ctr1 = Ctr1 + 1.

BingoCard = "Card 2 on Vertical: ".

BingoCard = BingoCard + Ctr1. Stop.

EndIf.

VAR\_A = CB3[Ctr1][0].

VAR\_B = CB3[Ctr1][1].

VAR\_C = CB3[Ctr1][2].

VAR\_D = CB3[Ctr1][3].

VAR\_E = CB3[Ctr1][4].

Bingo = P1(VAR\_A;VAR\_B;VAR\_C;VAR\_D;VAR\_E).

If(Bingo == Yes)

Ctr1 = Ctr1 + 1.

BingoCard = "Card 3 on Vertical: ".

BingoCard = BingoCard + Ctr1. Stop.

EndIf.

Loop.

If(Bingo == No)

For(Ctr1 = 0; Ctr1 < 5; Ctr1++)

VAR\_A = CB1[0][Ctr1].

VAR\_B = CB1[1][Ctr1].

VAR\_C = CB1[2][Ctr1].

VAR\_D = CB1[3][Ctr1].

VAR\_E = CB1[4][Ctr1].

Bingo = P1(VAR\_A;VAR\_B;VAR\_C;VAR\_D;VAR\_E).

If(Bingo == Yes)

Ctr1 = Ctr1 + 1.

BingoCard = "Card 1 on Horizontal: ".

BingoCard = BingoCard + Ctr1. Stop.

EndIf.

VAR\_A = CB2[0][Ctr1].

VAR\_B = CB2[1][Ctr1].

VAR\_C = CB2[2][Ctr1].

VAR\_D = CB2[3][Ctr1].

VAR\_E = CB2[4][Ctr1].

Bingo = P1(VAR\_A;VAR\_B;VAR\_C;VAR\_D;VAR\_E).

If(Bingo == Yes)

Ctr1 = Ctr1 + 1.

BingoCard = "Card 2 on Horizontal: ".

BingoCard = BingoCard + Ctr1. Stop.

EndIf.

VAR\_A = CB3[0][Ctr1].

VAR\_B = CB3[1][Ctr1].

VAR\_C = CB3[2][Ctr1].

VAR\_D = CB3[3][Ctr1].

VAR\_E = CB3[4][Ctr1].

Bingo = P1(VAR\_A;VAR\_B;VAR\_C;VAR\_D;VAR\_E).

If(Bingo == Yes)

Ctr1 = Ctr1 + 1.

BingoCard = "Card 3 on Horizontal: ".

BingoCard = BingoCard + Ctr1. Stop.

EndIf.

Loop.

EndIf.

If(Bingo == No)

VAR\_A = CB1[0][2].

VAR\_B = CB1[1][2].

VAR\_C = CB1[3][2].

VAR\_D = CB1[4][2].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 1 on Horizontal Middle.". Stop.

EndIf.

VAR\_A = CB2[0][2].

VAR\_B = CB2[1][2].

VAR\_C = CB2[3][2].

VAR\_D = CB2[4][2].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 2 on Horizontal Middle.". Stop.

EndIf.

VAR\_A = CB3[0][2].

VAR\_B = CB3[1][2].

VAR\_C = CB3[3][2].

VAR\_D = CB3[4][2].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 3 on Horizontal Middle.". Stop.

EndIf.

EndIf.

If(Bingo == No)

VAR\_A = CB1[2][0].

VAR\_B = CB1[2][1].

VAR\_C = CB1[2][3].

VAR\_D = CB1[2][4].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 1 on Vertical Middle.". Stop.

EndIf.

VAR\_A = CB2[2][0].

VAR\_B = CB2[2][1].

VAR\_C = CB2[2][3].

VAR\_D = CB2[2][4].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 2 on Vertical Middle.". Stop.

EndIf.

VAR\_A = CB3[2][0].

VAR\_B = CB3[2][1].

VAR\_C = CB3[2][3].

VAR\_D = CB3[2][4].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 3 on Vertical Middle.". Stop.

EndIf.

EndIf.

If(Bingo == No)

VAR\_A = CB1[0][0].

VAR\_B = CB1[0][4].

VAR\_C = CB1[4][0].

VAR\_D = CB1[4][4].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 1 on All Sides.". Stop.

EndIf.

VAR\_A = CB2[0][0].

VAR\_B = CB2[0][4].

VAR\_C = CB2[4][0].

VAR\_D = CB2[4][4].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 2 on All Sides.". Stop.

EndIf.

VAR\_A = CB3[0][0].

VAR\_B = CB3[0][4].

VAR\_C = CB3[4][0].

VAR\_D = CB3[4][4].

Bingo = P2(VAR\_A;VAR\_B;VAR\_C;VAR\_D).

If(Bingo == Yes)

BingoCard = "Card 3 on All Sides.". Stop.

EndIf.

EndIf.

Ctr = Ctr + 1.

LoopIf((Bingo == No) && (Ctr < 75)).

Say BingoCard.

Say "\n\nCard 1\nB\tI\tN\tG\tO\n".

For(X = 0; X < 5; X++)

For(Y = 0; Y < 5; Y++)

Say CB1[Y][X] , "\t".

Loop.

Say "\n".

Loop.

Say "\nCard 2\nB\tI\tN\tG\tO\n".

For(X = 0; X < 5; X++)

For(Y = 0; Y < 5; Y++)

Say CB2[Y][X] , "\t".

Loop.

Say "\n".

Loop.

Say "\nCard 3\nB\tI\tN\tG\tO\n".

For(X = 0; X < 5; X++)

For(Y = 0; Y < 5; Y++)

Say CB3[Y][X] , "\t".

Loop.

Say "\n".

Loop.

End.

Task Boolean P1:

Start

Var Boolean Pass is No.

If((A == Yes) && (B == Yes) && (C == Yes) && (D == Yes) && (E == Yes))

Pass = Yes.

EndIf.

Response Pass.

End.

Task Boolean P2:

Start

Var Boolean Pass is No.

If((A == Yes) && (B == Yes) && (C == Yes) && (D == Yes))

Pass = Yes.

EndIf.

Response Pass.

End.

Task String Letter:

Start

Var String Resp is "".

If(N>60)

Resp = "O".

Or(N>45)

Resp = "G".

Or(N>30)

Resp = "N".

Or(N>15)

Resp = "I".

Otherwise

Resp = "B".

EndIf.

Response Resp.

End.#