CMPS 401
Survey of Programming Languages

Programming Assignment #1
FORTRAN Language
On the Ubuntu Operating System

1. Write a FORTRAN program (P1.f) under the Ubuntu Operating System that will handle weights, lengths, acceleration and temperatures equivalents:
    - Convert Pounds to Kilograms and vice versa.
    - Convert Feet to Meters and vice versa.
    - Convert Fahrenheit to Celsius and vice versa.

2. The program should ask the user what type of conversion he/she desires.
3. Upon choosing the desired type of conversion, the program should display the conversion **without** terminating allowing the user an option to terminate or select another type.

Example:

```
Enter a conversion option (1-6 or 0 to exit):
--------------------------
(1) Pounds to Kilograms
(2) Kilograms to Pounds
(3) Feet to meters
(4) Meters to feet
(5) Fahrenheit to Celsius
(6) Celsius to Fahrenheit
(0) Exit this program
--------------------------
```

4. You need a CMPS401 account on the Ubuntu Operating System.
5. If you need help with the FORTRAN programming, go to the following links:
    - Using GNU Fortran (gfortran.pdf)
    - Professional Programmer's Guide to Fortran77 (book by Clive Page, prof77.pdf)
    - Fortran 90 Standard (ISO/IEC 1539 : 1991, N692.pdf)
    - FORTRAN 77 Reference
    - Fortran 90 Tutorial
    - Google Fortran 95 Handbook
    - Wikipedia Fortran 95 language features

6. The following instructions are used to edit, compile, and run your program

```
nano     P1.f       // Ctrl + x: Exit the editor
gfortran P1.f       // GNU Fortran compiler
./a.out             // Run your program
```

7. There are six Digital Fortran examples to assist you on this assignment.
    1) A simple program to run (TSimple.f)
    2) Test data types and variables (TVar.f)
    3) Test selection statements (TSel.f)
    4) Test loops (TLoop.f)
    5) Test subprograms (TSub.f)
    6) Other concerns: Test Read from users (TRead.f)

8. Your program assignment #1 consists of the following two files under your CMPS401 account **home directory**:
    1) Fortran program (P1.f)
    2) Executable file (a.out)

   Note: Your files on the Ubuntu Operating System will be checked and should not be modified after due date.

Useful UNIX Commands

Command	**Description**
man	help menu
nano	simple text editor
gcc	compiles your source code “gnu C compiler”
a.out	executes your program
ls –al	displays a long list of files “includes hidden files i.e. dot files”
pwd	prints working directory “pathname”
cd	changes directory
mkdir	creates a directory
rmdir	removes a directory
cp file1 file2	copies contents of file1 into file2
mv file1 file2	moves a file from one place to another, or change its name
rm	removes a file
more	displays a file’s contents
grep	searches for a specified pattern in a file or list of files
ps	obtains the status of the active processes in the system
kill –9 pid	terminates all processes
passwd	modify a user's password
logout	terminates your session
who	display who is on the system
finger	displays the user information
date > myfile	“output redirection” saves the output of date command in myfile
cal >> myfile	**“**appends” calendar to myfile
cal	display a calendar and the date
wc file1	counts the number of lines, words, and characters in file1

Examples

A Simple Program to Run (TSimple.f)

```
C Display "Hello" on your screen
C Program-ID:   TSimple.f
C Author:       Kuo-pao Yang
C OS:           Ubuntu 20
C Compiler:     GNU Fortran
C Note:
C The following instructions are used to
C        edit, compile, and run this program
C    $ nano     TSimple.f
C    $ gfortran TSimple.f
C    $ ./a.out

C The first 6 character positions on each line
C   are reserved for statement labels
      PROGRAM TSimple
C UNIT=* selects the standard output file which is
C   normally your own terminal;
C FMT=* selects a default output layout
      WRITE (UNIT=*,FMT=*) 'Hello'
C An abbreviated form
      WRITE (*,*) 'Hello'
      END
C Input:  No
C Output:
C       Hello
C       Hello
```

Data Types and Variables (TVar.f)

```
C Test Data types and variables
C 6 Data Types:
C   INTEGER: 32 bits
C   REAL:    32 bits (Single precision)
C   DOUBLE PRECISION: 64 bits
C   COMPLEX: a + bi (a and b single precision)
C   LOGICAL: .TRUE. or .FALSE.
C   CHARACTER: a char like 'A' or a string like 'BCD'
C Program-ID:   TVar.f
C Author:       Kuo-pao Yang
C OS:           Ubuntu 20
C Compiler:     GNU Fortran
C Note:
C The following instructions are used to
C       edit, compile, and run this program
C    $ nano     TVar.f
C    $ gfortran TVar.f
C    $ ./a.out

C The first 6 character positions on each line
C   are reserved for statement labels
      PROGRAM TVar
      INTEGER m
      REAL    n
      DOUBLE PRECISION o
      COMPLEX p
      LOGICAL q
      CHARACTER r
      CHARACTER s*3
      m = 1
      n = 2
      o = 3
      p = (4, 5)
      q = .FALSE.
      r = 'A'
      s = 'BCD'
      WRITE(*,*) 'm = ', m
      WRITE(*,*) 'n = ', n
      WRITE(*,*) 'o = ', o
      WRITE(*,*) 'p = ', p
      WRITE(*,*) 'q = ', q
      WRITE(*,*) 'r = ', r
      WRITE(*,*) 's = ', s
      END
C Input:  No
C Output:
C     m =            1
C     n =    2.0000000
C     o =    3.0000000000000000
C     p =  (  4.0000000    ,  5.0000000    )
C     q =  F
C     r = A
C     s = BCD
```

Selection Statements (TSel.f)

```
C Test Selections: if, if-else, nested if-else
C Logical Operators:    .AND., .OR., .NOT.
C Relational Operators:
C   .LT. (Less Than),  .GT. (Greater Than),  .EQ. (Equal to)
C   .LE. (Less Equal), .GE. (Greater Equal), .NE. (Not Equal)
C Program-ID:   TSel.f
C Author:       Kuo-pao Yang
C OS:           Ubuntu 20
C Compiler:     GNU Fortran
C Note:
C The following instructions are used to
C       edit, compile, and run this program
C    $ nano     TSel.f
C    $ gfortran TSel.f
C    $ ./a.out

      PROGRAM TSel
      INTEGER m, n, o, p, q, r
      m = 1
      n = 2
      o = 3
      p = 4
      q = 5
      r = 6
      IF (p .GT. m) THEN
        WRITE(*,*) 'p > m '
      ENDIF
      IF (q .LT. n) THEN
        WRITE(*,*) 'q < n '
      ELSE
        WRITE(*,*) 'q >= n '
      ENDIF
      IF ((m .NE. n) .AND. (o .GE. n)) THEN
        r = m + n
        WRITE(*,*) 'r = ', r
      ELSE
        r = m - n
        r = r * o + 3
        IF ((p .EQ. q) .OR. (q .NE. r)) THEN
          r = r / 2
          WRITE(*,*) 'r = ', r
        ENDIF
      ENDIF
      END
C Input:  No
C Output:
C     p > m
C     q >= n
C     r =            3
```

Loops (TLoop.f)

```
C Test Loops:  do-loop, if-goto-loop, nested loop
C Program-ID:   TLoop.f
C Author:       Kuo-pao Yang
C OS:           Ubuntu 20
C Compiler:     GNU Fortran
C Note:
C The following instructions are used to
C       edit, compile, and run this program
C    $ nano     TLoop.f
C    $ gfortran TLoop.f
C    $ ./a.out

      PROGRAM TLoop
      INTEGER i, j, m, n
      INTEGER A(3, 4)
C Do-Loop: The number of iterations is fixed at the beginning
C     DO label, loop-control-var = init-value, final-value, step-size
C        statements
C     label CONTINUE
      m = 1
      DO 10, i = 3, 6, 2
         m = m + i
10    CONTINUE
      WRITE (*,*) 'm = ', m
C if-goto-loop
      n = 1
      i = 3
20    IF (i .LE. 6) THEN
         n = n + i
         i = i + 2
         GOTO 20
      END IF
      WRITE (*,*) 'n = ', n
C nested Do-loop
      DO 40, i = 1, 3, 1
         DO 30, j = 1, 4, 1
           A(i,j) = i + j
30       CONTINUE
40    CONTINUE
C nested if-goto-loop
      i = 1
50    IF (i .LE. 3) THEN
         j = 1
60       IF (j .LE. 4) THEN
            WRITE (*,*) 'A (', i, ',', j, ') = ', A(i,j)
            j = j + 1
            GOTO 60
         ENDIF
         i = i + 1
         GOTO 50
      ENDIF
      END
```

```
C Input:  No
C Output:
C     m =            9
C     n =            9
C     A (           1,            1) =            2
C     A (           1,            2) =            3
C     A (           1,            3) =            4
C     A (           1,            4) =            5
C     A (           2,            1) =            3
C     A (           2,            2) =            4
C     A (           2,            3) =            5
C     A (           2,            4) =            6
C     A (           3,            1) =            4
C     A (           3,            2) =            5
C     A (           3,            3) =            6
C     A (           3,            4) =            7
```

Subprograms (TSub.f)

```
C Test Subprograms:  FUNCTION
C Program-ID:   TSub.f
C Author:       Kuo-pao Yang
C OS:           Ubuntu 20
C Compiler:     GNU Fortran
C Note:
C The following instructions are used to
C       edit, compile, and run this program
C    $ nano     TSub.f
C    $ gfortran TSub.f
C    $ ./a.out

      PROGRAM TSub
      INTEGER m, n
      INTEGER sub1, sub2
      m = sub1()
      WRITE(*,*) 'm = ', m
      m = 2
      n = 3
      n = sub2(m, n) + 4
      WRITE(*,*) 'n = ', n
      END

      FUNCTION sub1() RESULT (r)
        INTEGER r
        r = 1
      END
      FUNCTION sub2(p, q) RESULT (r)
         INTEGER p, q, r
         r = p + q
      END
C Input:  No
C Output:
C     m =            1
C     n =            9
```

Others (TRead.f)

```
C Test Subprograms:  READ (from users)
C Program-ID:   TRead.f
C Author:       Kuo-pao Yang
C OS:           Ubuntu 20
C Compiler:     GNU Fortran
C Note:
C The following instructions are used to
C       edit, compile, and run this program
C    $ nano     TRead.f
C    $ gfortran TRead.f
C    $ ./a.out

      PROGRAM TRead
      INTEGER m
      REAL    n
      WRITE(*,*) 'Enter an integer value for m'
      READ(*,*) m
      WRITE(*,*) 'm = ', m
      WRITE(*,*) 'Enter a real value for n'
      READ(*,*) n
      WRITE(*,*) 'n = ', (n / 2)
      END
C Input:  User Inputs
C Output:
C     Enter an integer value for m
C     5
C     m =            5
C     Enter a real value for n
C     3.43
C     n =    1.71500003
```