```
 1:
 2: ****************************************
 3: ./exercise4/mclau.f90
 4:
 5: PROGRAM mclau
 6: !** Program to calculate SIN(x) to ‘n’ terms of the Mclaurin Series
 7: !** NOTE
 8: !**      Limit is at 10**38 for REAL type that is
 9: !**
10: !**  factorial           34  =   2.9523282E+38
11: !**  factorial           35  =  Infinity    occurs at term 18
12: !**  NOTE compiler clever and reworks math in loop to avoid REAL limitation here
13: !**  HOWEVER Not as clever as our mclau2.f90 code
14:
15:   IMPLICIT NONE
16:
17:   INTEGER :: i,sign=-1,n
18:   REAL :: sinx,x,fact=1
19:
20:   PRINT*,’Enter the value of x you require: ’
21:   READ*,x
22:   PRINT*,’Enter the number of terms you require: ’
23:   READ*,n
24:
25:   sinx=x
26:   PRINT*,’Calculating Term’,1
27:
28:   DO i=3,2*n-1,2
29:   !** Loop through starting with the second term (-x^3/3!) ans stopping
30:   !** loops up to 2*n-1 for the last term.
31:
32:     PRINT*,’Calculating Term’,(i-1)/2+1
33:     fact=fact*i*(i-1)        ! *** Calculate to rquired factorial
34:     sinx=sinx+sign*x**i/fact ! *** Calculate the new approximation
35:     sign=-sign               ! *** update the sign parameter
36:   END DO
37:
38:   PRINT*,’The approximation is for x=’,x
39:   PRINT*,’Number of terms in the approximation = ’,n
40:   PRINT*,’The approximation =’,sinx
41:   PRINT*,’The true value is =’,SIN(x)
42:
43: END PROGRAM mclau
44:
45: ****************************************
46:
47:
48: ****************************************
49: ./exercise4/mclau2.f90
50:
51: PROGRAM mclau2
52:
53: !**
54: !** Program to calculate SIN(x) to ‘n’ terms of the Mclaurin Series
55: !** sin(x)=x-x^3/3!+x^5/5!-x^7/7!+x^9/9! ..............
56: !**
57: !** Version that does not suffer from factoral calculation limits
58: !**
59:
60:   IMPLICIT NONE
61:
62:   INTEGER :: i,sign=-1,n
63:   REAL :: sinx,x,dummy
64:
65:   PRINT*,’Enter the value of x you require: ’
66:   READ*,x
67:   PRINT*,’Enter the number of terms you require: ’
```



```
68:   READ*,n
69:
70:   sinx=x
71:   dummy=x
72:
73:   PRINT*,’Calculating Term’,1
74:
75:   DO i=3,2*n-1,2
76:   !** Loop through starting with the second term (-x^3/3!) ans stopping
77:   !** loops up to 2*n-1 for the last term.
78:     PRINT*,’Calculating Term’,(i-1)/2+1
79:     dummy=dummy/(i*(i-1))*x**2  ! *** Calculate to rquired dummyorial
80:     sinx=sinx+sign*dummy        ! *** Calculate the new approximation
81:     sign=-sign                  ! *** update the sign parameter
82:   END DO
83:
84:   PRINT*,’The approximation is for x=’,x
85:   PRINT*,’Number of terms in the approximation = ’,n
86:   PRINT*,’The approximation =’,sinx
87:   PRINT*,’The true value is =’,SIN(x)
88:
89: END PROGRAM mclau2
90:
91: ****************************************
92:
93:
94: ****************************************
95: ./exercise2/quad_complex.f90
96:
97: PROGRAM quad_complex
98:
99: !*** Program to calculate the roots of a quadratic
100: !*** usingthe more general  COMPLEX data type
101:
102:   IMPLICIT NONE
103:
104:   REAL    :: a,b,c
105:   COMPLEX :: root1,root2,sqdiscrim
106:   LOGICAL :: check
107:
108:
109:   ! ** Enter the coefficients a,b,c
110:   PRINT*,"Enter A, b, and c of the &
111:             &polynomial ax**2 + bx + c:"
112:
113:   READ*,a,b,c  !*** Read in coefficients from keyboard
114:
115:   check= .NOT. a==0  !** Set check to .TRUE. if a valid quadratic
116:
117:   IF (check) THEN
118:
119:     !** Calculate the sqrt of discriminant as a complex number
120:     sqdiscrim=SQRT(CMPLX(b**2 - 4.0*a*c))
121:
122:     root1=(-b + sqdiscrim)/(2.0*a) !** Calculate root1
123:     root2=(-b - sqdiscrim)/(2.0*a) !** Calculate root2
124:
125:     PRINT*,’The roots are:’
126:     PRINT*,"Root1 : ","Real Part=",REAL(root1),&
127:              " : Imaginary part=",AIMAG(root1)
128:     PRINT*,"Root2 : ","Real Part=",REAL(root2),&
129:              " : Imaginary part=",AIMAG(root2)
130:   ELSE
131:     PRINT*,"This is not a valid quadratic"
132:   ENDIF
133:
134: END PROGRAM quad_complex
```

```
  135:
  136:
  137: ****************************************
  138:
  139:
  140: ****************************************
  141: ./exercise3/isotope.f90
  142:
  143: PROGRAM isotope
  144: !**
  145: !** Code to calculate the "time passed" since death of a once living being.
  146: !**
  147:
  148:   IMPLICIT NONE
  149:
  150:
  151:   REAL time,   & !*** Estimation of current age
  152:        percent   !*** Percent of isotope remaining in sample
  153:
  154:   !*** Carbon14 decay constant
  155:   REAL, PARAMETER :: lambda=0.00012097
  156:
  157:
  158:   PRINT*,"Please input the percentage of carbon 14 remaining"
  159:   READ*,percent
  160:
  161:
  162:   time=-log(percent/100)/lambda
  163:
  164:
  165:   PRINT*,"Age estimation of sample in years is : ",time
  166:
  167:
  168: END PROGRAM isotope
  169: ****************************************
  170:
  171:
  172: ****************************************
  173: ./exercise1/integer_test.f90
  174:
  175: PROGRAM integer_test
  176: !**
  177: !** Program to highlight problems in integer arithmetic
  178: !**
  179:
  180:   IMPLICIT NONE
  181:
  182:   INTEGER :: a=1, b=2,answer1
  183:   REAL :: answer2,answer3
  184:
  185:   answer1=a/b
  186:   answer2=a/b
  187:   answer3=REAL(a)/b
  188:
  189:   PRINT*,"When evaluated as integer =",answer1
  190:   PRINT*,"When assigned to a real   =",answer2
  191:   PRINT*,"When a is casted first    =",answer3
  192:
  193: END PROGRAM integer_test
  194:
  195: ****************************************
  196:
```