1. Write a C function that returns a constant of type Q32, computed from the ratio of two integers without using any floating-point operations. Write a C program to test your function. The function prototype is:

```
Q32 Q32Ratio(int32_t dividend, int32_t divisor) ;
```

```
typedef int64_t Q32 ;

Q32 Q32Ratio(int32_t top, int32_t btm)
      {
      int32_t rounding = (((top ^ btm) >= 0) ? btm : -btm) / 2 ;
      return  (((Q32) top << 32) + rounding) / btm ;
      }
```

2. Write a C function that prints a Q32 number as a real number without using any floating-point operations. Write a C program to test your function. The function prototype is:

```
void PrintQ32(Q32) ;
```

```
#define Upper32Bits(x) ((uint32_t *) &x)[1]

void PrintQ32(Q32 x)
      {
      int k ;

      if (x < 0) { putchar('-') ; X = -x ; }
      printf("%u", Upper32Bits(x)) ;
      putchar('.') ;
      for (k = 0; k < 5; k++)
            {
            Upper32Bits(x) = 0 ;
            x = 10 * x ;
            putchar('0' + Upper32Bits(x)) ;
            }
      }
```

3. Use the function developed in problem 1 to create two Q32 values, 12.34 and -56.78 and write a C program compute and display their sum and difference.

```
typedef int64_t Q32 ;

int main(void)
      {
      Q32 one = Q32Ratio(1234, 100) ;
      Q32 two = Q32Ratio(-5678, 100) ;
      printf(" sum = ") ; PrintQ32(one + two) ; putchar('\n') ;
      printf("diff = ") ; PrintQ32(one – two) ; putchar('\n') ;
      return 0 ;
      }
```

4. Write a C function to calculate the area of a circle using Q32 fixed-point reals and the multiply and divide routines of Listing 10-1 and Listing 10-4. Write a C program to test your function. The function prototype is:

```
Q32 CircleArea(Q32 radius) ;
```

```
Q32 CircleArea(Q32 radius)
      {
      Q32 pi = Q32Ratio(314159, 100000) ;
      Q32 rSquared = Q32Product(radius, radius) ;
      return Q32Product(pi, rSquared) ;
      }
```

5. Write a C function to compute the dot product of two vectors using Q32 fixed-point reals and the multiply and divide routines of Listing 10-1 and Listing 10-4. Write a C program to test your function. The function prototype is:

```
Q32 DotProduct(Q32 vec1[],Q32 vec2[], int32_t len) ;
```

```
Q32 DotProduct(Q32 vec1[],Q32 vec2[], int32_t len)
   {
   Q32 result ;
   int k ;

   result = 0 ;
   for (k = 0; k < len; k++)
      {
      result += Q32Product(vec1[k],vec2[k]) ;
      }
   return result ;
   }
```

6. Write a C function to evaluate a polynomial using Q32 fixed-point reals and the multiply and divide routines of Listing 10-1 and Listing 10-4. Write a C program to test your function. The function prototype is:

```
Q32 Polynomial(Q32 x, Q32 coef[], int32_t terms) ;
```

```
Q32 Polynomial(Q32 x, Q32 coef[], int32_t terms)
   {
   Q32 result ;
   int k ;

   result = 0 ;
   power = Q32Ratio(1, 1) ;
   for (k = 0; k < terms; k++)
      {
      result += Q32Product(coef[k], power) ;
      power = Q32Product(power, x) ;
      }
   return result ;
   }
```

7. Write a function in C that calls the Polynomial function developed in problem 6 to compute an eight term Taylor series approximation to the inverse, X^{-1}. (Do not use a divide!) Write a C program to test your function. The function prototype is:

```
Q32 Inverse(Q32 x) ;
```

Note: $x^{-1} = (x - 1)^0 - (x - 1)^1 + (x - 1)^2 - (x - 1)^3 + \ldots$

```
Q32 Inverse(Q32 x)
      {
      Q32 one = Q32Ratio(1, 1) ;
      static Q32 coef[8] ;
      static int initialize = 1 ;
      int k, sign ;

      if (initialized)
            {
            for (k = 0; k < 8; k++)
                  {
                  coef[k] = (k % 2) ? -one : one ;
                  }
            initialize = 0 ;
            }
      return Polynomial(x - one, coef, 8) ;
      }
```

8. Write a function in C that calls the Polynomial function developed in problem 6 to compute an eight term Taylor series approximation to the trigonometric sine. Write a C program to test your function. The function prototype is:

```
Q32 Sine(Q32 radians) ;
```

Note: $\text{sine}(x) = x^1/1! - x^3/3! + x^5/5! - x^7/7! + \ldots\ldots$

```
Q32 Sine(Q32 radians)
      {
      static Q32 coef[8] ;
      static int initialize = 1 ;
      int32_t top, btm ;
      int k ;

      if (initialize)
            {
            top = btm = 1 ;
            for (k = 0; k < 8; k++)
                  {
                  coef[k] = (k % 2) ? Q32Ratio(top, btm) : 0 ;
                  if (k % 2) top = -top ;
                  btm *= k + 1 ;
                  }
            initialize = 0 ;
            }
      return Polynomial(x, coef, 8) ;
      }
```

9. Write a function in C that calls the Polynomial function developed in problem 6 to compute an eight term Taylor series approximation to the exponential. Write a C program to test your function. The function prototype is:

   ```
   Q32 Exponential(Q32 x) ;
   ```

   Note: $e^x = x^0/0! + x^1/1! + x^2/2! + x^3/3! + x^4/4! + \ldots$

```
Q32 Exponential(Q32 x)
      {
      static Q32 coef[8] ;
      int initialize = 1 ;
      int32_t btm ;
      int k ;

      if (initialize)
            {
            btm = 1 ;
            for (k = 0; k < 8; k++)
                  {
                  coef[k] = Q32Ratio(1, btm) ;
                  btm *= k + 1 ;
                  }
            initialize = 0 ;
            }
      return Polynomial(x, coef, 8) ;
      }
```