5.34- ***(Recursive Exponentiation)*** Write a recursive function power(base, exponent)that when invoked returns

$$base^{exponent}$$

For example, $power(3,4) = 3 * 3 * 3 * 3$. Assume that exponentis an integer greater than or equal to 1. $Hint$: The recursion step would use the relationship

$$base^{exponent} = base * base^{exponent-1}$$

and the terminating condition occurs when exponent is equal to 1because $base^1 = base$.

ANSWER

```
#include<stdio.h>

int perfect(int x);

int exponentiate(int base, int exponent) {
   if(exponent== 0) {
      return 1;
   } else {
      return base * exponentiate(base, exponent - 1);
   }
}
```

5.30- (***Quality Points for Student's Grades***) Write a function $qualityPointsthat$ inputs a student's average and returns 4 it's 90–100, 3 if it's 80–89, 2 if it's 70–79, 1 if it's 60–69, and 0 if the average is lower than 60.

ANSWER

```
#include <stdio.h>

int qualityPoints(int x);

int main()
{
   int x;
   printf("Please enter the student's average: ");
   scanf("%d", &x);
   printf("%d\n", qualityPoints(x));
   return 0;
}

int qualityPoints(int x)
{

   if (x >= 90 && x <= 100)
            return 4;
   else if (x >= 80 && x <= 89)
            return 3;
   else if (x >= 70 && x <= 79)
            return 2;
   else if (x >= 60 && x <= 69)
            return 1;
   else if (x >= 0 && x < 60)
            return 0;
   else                              /* a little bit of checking */
            return -1;               /* wrong result!. eg mean >100 or

}
```

5.41- (***Distance between Points***) Write a function distance that calculates the distance between two points *(x1, y1)* and *(x2, y2)*. All numbers and return values should be of type double.

ANSWER

```
#include <stdio.h>
#include <math.h>

double distance(double x1, double y1, double x2, double y2)
{
   double square_difference_x = (x2 - x1) * (x2 - x1);
   double square_difference_y = (y2 - y1) * (y2 - y1);
   double sum = square_difference_x + square_difference_y;
   double value = sqrt(sum);
   return value;
}
```

Free- (***Convert to Binary***) Write a program in C to convert decimal number to binary number using the function.

ANSWER

```
#include<stdio.h>

long toBin(int);

int main()
{
   long bno;
   int dno;
  printf("\n\n Function : convert decimal to binary :\n");
   printf("-------------------------------------------\n");
   printf(" Input any decimal number : ");
   scanf("%d",&dno);
   bno = toBin(dno);
   printf("\n The Binary value is : %ld\n\n",bno);

   return 0;
}
long toBin(int dno)
{
   long bno=0,remainder,f=1;
   while(dno != 0)
   {
       remainder = dno % 2;
       bno = bno + remainder * f;
       f = f * 10;
       dno = dno / 2;
   }
   return bno;
}
```

Free- ***(Lowest Common Multiple)*** Write a C program to find LCM of two numbers using recursion. How to find LCM of two numbers in C programming using recursion. Logic to find LCM of two numbers using recursion.

ANSWER

```
#include <stdio.h>

/* Function declaration */
int lcm(int a, int b);

int main()
{
   int num1, num2, LCM;

   /* Input two numbers from user */
   printf("Enter any two numbers to find lcm: ");
   scanf("%d%d", &num1, &num2);

   /*
    * Ensures that first parameter of LCM function
    * is always less than second
    */
   if(num1 > num2)
      LCM = lcm(num2, num1);
   else
      LCM = lcm(num1, num2);

   printf("LCM of %d and %d = %d", num1, num2, LCM);

   return 0;
}


/**
 * Recursive function to find lcm of two numbers 'a' and 'b'.
 * Here 'a' needs to be always less than 'b'.
 */
int lcm(int a, int b)
{
   static int multiple = 0;

   /* Increments multiple by adding max value to it */
   multiple += b;

   /*
    * Base condition of recursion
    * If found a common multiple then return the multiple.
    */
   if((multiple % a == 0) && (multiple % b == 0))
   {
      return multiple;
   }
   else
   {
      retur
n lcm(a, b);
   }
}
   return bno;
}
```