/* Coding Problems (25 points) */

/*

Please write code for the two problems below. Write the code for both problems in functions below the main function and call both from the main function. The functions should accept data from main, calculate and return the result to main. Getting data from the user and printing the results should be done in main. When completed, the main function should get input data from the user, call the functions, receive the results of the function calls, and print their results to screen.

1. You have saved $500 to use as a down payment on a car. Before beginning your car shopping, you decide to write a function to help you figure out what your monthly payment will be, given the car's purchase price, the monthly interest rate, and the time period over which you will pay back the loan. The formula for calculating your payment is:

$$\text{payment} = \frac{iP}{1 - (1 + i)^{(-n)}}$$

where
P = principal (the amount you borrow)
i = monthly interest rate (1/12 of the annual rate)
n = total number of payments

Your program should get input data from the user in the main function in this manner:

```
CALCULATE PAYMENT
Enter principal: 500.0
Enter annual interest rate: 6.0
Enter number of payments: 24
Your monthly payment is: 22.16
```

2. For any integer n > 0, n! is defined as the product n * (n - 1) * (n - 2) * 2 * 1.
0! is defined to be 1. It is sometimes useful to have a closed-form definition instead;
for this purpose, an approximation can be used. R.W. Gosper proposed the following such approximation formula:

```
      n! = n^n x e^(-n) x square_root((2n + 1/3) x pi)

   Create a function that prompts the user to enter an integer n , uses
Gosper's
   formula to approximate n!, and then displays the result. The input data
should
   be read and output result should be printed in the main function, and the
calculation
   should be in the function for Problem 2. The message displaying the result
should look
   something like this:

      Enter a number: 5
      5! equals approximately 119.97003

   Your program will be easier to debug if you use some intermediate values
   instead of trying to compute the result in a single expression. If you are not
   getting the correct results, then you can compare the results of your
intermediate
   values to what you get when you do the calculations by hand. Use at least
two
   intermediate variables - one for 2n + 1/3 and one for (2n + 1/3) x pi.

   Display each of these intermediate values to simplify debugging. Be sure to
   use a named constant for PI, and use the approximation 3.14159265 and
use a
   constant for E, approximated as 2.718281827. Test the program on
nonnegative
   integers less than 8.

*/

// Preprocessor directives
#include <stdio.h>
#include <math.h>
#define PI 3.14159265
#define EXP 2.718281827
// Function prototypes
double CalculatePayment(double PrincipalAmount, double
AnnualInterestRate, int NumberofPayments);
double GosperFactorial(int number);
// Main function
int main()
{
// CALCULATE PAYMENT (Problem 1)
```

```
double UserPrincipalAmount = 0.0;
double UserAnnualInterestRate = 0.0;
int UserNumberofPayments = 0;
printf("This program calculates your monthly payment for your loan \n");
printf("Please enter your principal amount:");
scanf("%lf", &UserPrincipalAmount);
printf("\n Please enter your annual interest rate:");
scanf("%lf", &UserAnnualInterestRate);
printf("\n Please enter your number of payments:");
scanf("%d", &UserNumberofPayments);

double UserMonthlyPayment = CalculatePayment(UserPrincipalAmount,
UserAnnualInterestRate, UserNumberofPayments);
printf("Your monthly payment due is: %f", UserMonthlyPayment);

// Gosper Method (Problem 2)
int UserNumber = 0;
printf("Please enter a number:");
scanf("%d", &UserNumber);

double UserGosperResult = GosperFactorial(UserNumber);
printf("%d! equals approximately %lf", UserNumber, UserGosperResult);

return 0;
}

// Function for Coding Problem 1
double CalculatePayment(double PrincipalAmount, double
AnnualInterestRate, int NumberofPayments)
{
  double MonthlyInterestRate = (AnnualInterestRate/100.00)/12.00;
  double PaymentAmount = (MonthlyInterestRate * PrincipalAmount)/(1 -
pow((1 + MonthlyInterestRate), (-1 * NumberofPayments)));
  return PaymentAmount;
}


// Function for Coding Problem 2
//Gosper Factorial
double GosperFactorial(int number)
{
double FactorialByGosperMethod = 0.0;

double IntermediateVariable1 = (2 * number) + (1.0/3.0);
//printf("\n Value of Intermediate Variable 1 = %lf", IntermediateVariable1);
```

```
double IntermediateVariable2 = (IntermediateVariable1 * PI);
//printf("\n Value of Intermediate Variable 2 = %lf", IntermediateVariable2);

double NPowerN = pow(number,number);
//printf("\n Value of NPowerN = %lf", NPowerN);

double EPowerNegN = pow(EXP, (-1 * number));
//printf("\n Value of EPowerNegN = %lf", EPowerNegN);

double SquareRootofIntermediate = sqrt(IntermediateVariable2);
//printf("\n Value of SquareRootofIntermediate = %lf",
SquareRootofIntermediate);

FactorialByGosperMethod = NPowerN * EPowerNegN *
SquareRootofIntermediate;
return FactorialByGosperMethod;

}
```