CS19001 Programming and Data Structures Lab

Assignment Set 5 **Prof. Pallab Dasgupta** **Feb 3, 2020**

---

1. Write a recursive function, **`ipow(x,n)`**, to return the value of $\mathbf{x^n}$, where ***n*** is a non-negative integer, using repeated squaring, that is:

$$x^{2n} = (x^n) \times (x^n) \qquad \text{and} \qquad x^{2n+1} = (x) \times (x^{2n})$$

   (a) Write a main program that reads a floating point number, x, and an integer, n, and calls the function to return and print the nth power of x.

   (b) In the function use a global variable, **count**, to count the number of multiplications performed.

2. You are given a fair dice and asked to compute the probability of having *k* sixes in *n* rolls of the dice. The probability is given by the Binomial term:

$$P(k,n) = {}^{n}C_{k} p^{k} (1-p)^{n-k}$$

   The probability of getting a six in a single roll of the dice is *p* = 1/6. The probability of getting at most *k* sixes in *n* rolls of the dice is given by:

$$P(\leq k,n) = \sum_{j=0}^{k} {}^{n}C_{j} p^{j} (1-p)^{n-j}$$

   (a) For computing P(≤k, n), we need the value of ${}^{n}C_{j}$ while computing the j^{th} term of the summation. We know that ${}^{n}C_{j} = \frac{n-j+1}{j} {}^{n}C_{j-1}$ and therefore it is easy to compute ${}^{n}C_{j}$ from ${}^{n}C_{j-1}$ which was anyway computed for the $(j–1)^{th}$ term. Write a function, ***getterm,*** which returns the value of ${}^{n}C_{0}$ when called the first time, ${}^{n}C_{1}$ when called the second time, ${}^{n}C_{2}$ when called the third time (use a static variable).

   (b) Write a function for computing P(≤k, n) using the function ***getterm*** and a main( ) to read the values of *k* and *n* and print the value of P(≤k, n).

3. **Polynomials.** A polynomial $a_0 + a_1x + a_2x^2 + \ldots + a_kx^k$ of degree *k* can be represented by a single dimensional array, A[ ], of *k+1* floating pointing numbers, where $A[j] = a_j$. Write the following functions in C:

Function Prototype	**Description**
void read_poly(FILE *fp, float A[ ], int k)	Reads coefficients of a polynomial of degree k from a file into array A
float eval_poly(float A[ ], int k, float x)	Returns the value of polynomial A for given value of x
void add_poly(float A[ ], float B[ ], float C[ ], int k)	Adds polynomials A and B into C
void mul_poly(float A[ ], float B[ ], float C[ ], int k)	Multiplies polynomials A and B into C
void print_poly(float A[ ], int k)	Prints the polynomial

Write a program, **asg11.c**, which does the following:

(a) It opens a file, input.dat, using the following code:

```
FILE *fp, *fopen();
fp = fopen(“input.dat”, “r”);
if (fp == NULL) { printf(“Unable to open file.\n”);
                      exit(0); }
```

(b) It reads the value of *k* (assume that it is always less than 10) from the file.

(c) It uses the function read_poly( ) to read polynomials A and B of degree *k* from the file.

(d) It uses the function add_poly( ) to find the polynomial C representing the sum of A and B

(e) It uses the function mul_poly( ) to find polynomial D representing the product of A and B.

(f) It uses the function print_poly( ) to print the polynomials, A, B, C, and D into the terminal.

(g) It reads a value of *x* from the terminal.

(h) It uses eval_poly( ) to compute the values of the polynomials, A, B, C, and D. These values are then printed into the terminal.

For the polynomials, $p(x) = 3x^4 + 5x^2 – 7.5x + 20$ and $q(x) = 8x^4 + 9.2x^3 – 14$, the sample format of the input file is as follows (the first line has the value of *k*):

```
4
20 -7.5 5 0 3
-14 0 0 9.2 8
```

4. The ministry of magic produces coins of denomination 3, 5 and 10 respectively. The function, **`canchange(k)`**, returns –1 if it is not possible to pay a value of *k* using these coins. Otherwise it returns the ***minimum*** number of coins needed to make the payment.

   For example, **`canchange(7)`** will return –1. On the other hand, **`canchange(14)`** will return 4 because 14 can be paid as 3+3+3+5 and there is no other way to pay with fewer coins.

   A code skeleton for the function is given below as a hint. This is not complete, and has missing statements and missing expressions, indicated with question marks.

```
int canchange(int k)
{
   int a= ?? ;
   if (k==0) return 0;
   if ( ?? ) return 1;
   if (k < 3) ??;

  a = canchange( ?? );
  if (a > 0) return ?? ;

  a = canchange(k – 5);
  if (a > 0) return ?? ;

  a = canchange( ?? );
  if (a > 0) return ?? ;
  ??
}
```

   (a) Complete the function and write a main( ) to read an input number, call the function with it, and print the value it returns.

   (b) Modify the function of part (a) to write a function to print the change. For example, if we call the function **`printchange(14)`** it should print 3+3+3+5. The function prototype is:

```
int printchange(int k)
```