AMAT 2120 — Fall 2005
Assignment 4 — Due Monday Nov.14, 2005

Part B should be submitted by those students whose midterm **programming question** mark is **20 or less** and by those whose **overall** mark is **57 or less**.

Other students need not submit that part and if they choose to work it out, it won't be marked.

Those students who are required to submit part B are welcome to submit part A as well, but it will be evaluated only if part B mark is 70% or more. In that case, an extra credit towards the midterm results may be given.

All students: Please don't forget also to submit your programs electronically.

A1 Write the midterm summation program using a wrap funciton `mySin` instead of the standard `sin` of `math.h`. I ask you to write three versions of the function `mySin`: one in the ANCI C style, `double mySin(double x)`. Another one passing the result by reference in C++ style, `void mySin(double x, double & result)`. The last one passing the result by address: `void mySin(double x, double * result)`. For all three versions, use the appropriate format of the function call in `main()`. If you want to avoid having three different files for your three versions, use the `#define` and `#ifdef-#else` preprocessor directive to switch between versions.

A2 Write a C program that evaluates the given polynomial

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n$$

at the given value of x. The values of n (the degree), x (the arguments) and the coefficients $a_0, \ldots, a_n$ should be provided by the user. It is up to you to decide which form of communication with users your program will employ: interactive input, command line arguments or input from a file. Use an array to store the coefficients a_k.

B1 Write a C program that implements a Horner-like method of summation of the geometric progression

$$S = b_0 + b_1 + \ldots + b_{n-1}, \quad \text{where} \quad b_n = b_0 q^{n-1},$$

b_0, q, N are given. The method I want you to use here is based on the possibility to rewrite the series following this pattern (for $N = 4$):

$$b_0 + b_0 q + b_0 q^2 + b_0 q^3 = b_0(1 + q(1 + q(1 + q))).$$

In general, you need to initialize the accumilator variable (say, `s`) with value 1. Then at each step you multiply the accumulator by q and add 1, obtaining in succession

$$1, \quad q+1, \quad q(q+1)+1, \quad q(q(q+1)+1)+1, \ldots.$$

At the end, after the necessary number of steps, don't forget to multiply the accumulated result by b_0 to get the final answer.

Do not use arrays in your program; it is unnecessary and irrelevant.

Make your program accept values from the user interactively, make sure that it is user-friendly. Some degree of robustness is desirable, too.

A good idea is also to include comparison of the computed result with closed-form expression for the geometric sum.

B2 Trace the following program and determine the output. Pay attention to exact details of the loop; it is not a standard counting loop!

```
int main()
{
  int i, j, rem;
  j=40;
  printf("Before loop, j=%d\n", j);
  for (i=1; i<=j; j--)
  {
    rem=j%i;
    j=j/i;
    if (rem!=0)
    {
      i=rem;
      printf("Case 1: remainder is nonzero\n");
    }
    else
    {
      i=i+1;
      printf("Case 2: remainder is zero\n");
    }
    printf("i=%d, j=%d\n", i,j);
  }
  printf("After loop: i=%d, j=%d\n", i,j);
  return(0);
}
```