CMPT 102 SAMPLE FINAL SOLUTIONS

PART 1: SHORT ANSWER QUESTIONS: TOTAL MARKS 40/100

1. Give a brief algorithm, outlining the following methods
   a. (5 points) Horner's algorithm for evaluating a polynomial
   b. (8 points) Newton Raphson method of finding the root of a polynomial

2. (8 points) Illustrate the insertion sort algorithm using the following array as example data. Show step by step what happens when the following data array is sorted into ascending order using the insertion sort algorithm. At each step show the array and give a one or two sentence explanation of what has been done to the array. **DO NOT WRITE CODE OR A DETAILED ALGORITHM.**
Array of sample data (35, 23, 8, 17, 55)

Example steps (from some other algorithm):
Step 1: (23, 35, 8, 17, 55, 35) correctly order elements 0 and 1
Step 2: (23, 8, 35, 17, and 55 35) correctly order elements 1 and 2

3. (6 points) Based on the following variable declarations and initializations state the value of each expression or state why the expression is not valid

```
int x[2][5]= { (1,2,3,4},(10,22,12,14,16) };
char ch[14]="hello to you";
char *chp;
int *x1p, *xxp;
chp = &ch[3];
x1p=&x[1][3];
```

examples
*(chp+4) The value of the expression is 'o'
ch[14] Invalid. This is the 14th element in an array with 13 elements

a. x[0][4]
b. ch[12]
c. *(chp+12)
d. xxp
e. *x1p
f. ch[5]

4. (10 points) Write a recursive function to evaluate N! = 1*2*3*…*N. **Do not write a main program.** The prototype of your recursive function should be
   double fact(int N);

5. (3 points) Define a structure DataCount with 5 elements, an integer counter Count, double data values Data1, Data2, and Data 3, and a pointer to a structure of type node Nodep.

PART 2: PROGRAMMING PROBLEMS: TOTAL MARKS 60/100

6. (20 points) A statistics library containing the following functions is available for your use. To access these functions you must include <statistics.h>.

   The functions in the statistics library are as follows:
   float DataMean(float DataArray[], int DataArrayLength, int StartInd, int EndInd);
   float DataMedian(float DatArray[], int DataArrayLength, int StartInd, int EndInd);
   float DataMin(float DataArray[], int DataArrayLength, int StartInd, int EndInd);
   float DataMax((float DataArray[], int DataArrayLength, int StartInd, int EndInd);

   Each of these function takes an array of data containing DataArrayLength data points and uses the subset of those data points beginning with DataArray[StartInd] and ending with DataArray[EndInd] (both these points are included) to calculates the appropriate statistical value. It is the responsibility of the user to assure that elements StartInd and EndInd are both within the array DataArray of length DataArrayLength.

   Write a program using the statistics library. Your program should
   a. read a data file, statsin.dat, containing 500 floating point data values into an array XData declared to have 700 elements. You may assume that the data file is error free and contains the correct number of values.
   b. Calculate the sum, mean, minimum, maximum, and median of the data points XData[0] to XData[99]
   c. Calculate the mean and maximum of data points XData[100] to XData[499]
   d. Print the means calculated in parts b and c along with explanatory messages

   DO NOT WRITE CODE FOR THE FUNCTIONS IN THE STATISTICS LIBRARY !!!!!

7. (20 points) Write two functions with the following prototypes. The first function determines the sum, and difference of two 2-D arrays of fixed size. The second function prints an array of fixed size to the standard output (screen)

void SumDif(double InData1[6][4], double InData2[6][4],
double SumData[6][4], double DifData[6][4]);
void MatPrint(double mat1[6][4]);

The sum of two 2-D arrays A and B is defined to be
SumData[i][j]=A[i][j]+B[i][j]; for each i and j in the array
The difference of two 2-D arrays A and B is defined to be
DifData[i][j]=A[i][j]-B[i][j]; for each i and j in the array

Also write a main program to call your functions.

Assume you have a function, MatInit, with the prototype below. MatInit fills your array with random values. You should use MatInit to fill your input arrays with data, you should not read data from the screen or from a file.
`void MatInit(double mat1[6][4]);`

8. (20 points) Write a function to determine the root of a polynomial using the bisection method. The function should have the prototype below. .

double BisectPoly(double Coefs[], double IntervalEnd1,
double IntervalEnd2, int Order, int *NumIterp);

The order of the polynomial will be passed into the function as the argument ***Order***. The ends of the initial interval for bisection are passed into the function as arguments ***IntervalEnd1*** and ***IntervalEnd2***. The coefficients of the polynomial are passed to the function as the argument ***Coefs[]***. The number of iterations taken to find the root is returned as ****NumIterp***. The array ***Coefs*** contains the coefficients, ***Coefs[0]*** contains A_0, ***Coefs[1]*** contains A_1 and so on. The polynomial can be expressed in equation form as
$y = A_0 + A_1 X + \cdots + A_N X^N$

The bisection algorithm is an iterative algorithm (the same procedure is repeated many times, once per iteration). Each iteration consists of the following steps

a. Evaluate the polynomial at the endpoints (start and end) of the interval
b. Check to see if either endpoint is the root of the polynomial
c. Find the midpoint of the interval (start to end)
d. Determine which half of the interval contains a root.
e. The interval is redefined to be the half of the previous interval containing the root.

Assume the iterations stop when the length of the interval is less than 0.001.

You may assume that a function PolyEval with prototype

double PolyEval(double coeffs[], double X, int order);

is provided. This function evaluates the polynomial at X and returns the value of the polynomial as the value of the function. **DO NOT WRITE CODE** for function PolyEval.