NAME:

10.001 Introduction to Computer Methods
Midterm Examination
October 28, 1999

NAME: SOLUTION

Instructions:
Open book, Open notes.
Please answer all questions in space provided.

Allocate time for each problem based on point value for each problem.
If stuck on a problem, move and and come back to it!

Show all work as partial credit will be awarded.

Write name on each sheet.

Good Luck!

NAME:

1. (15 points)

Write the output of the following program.

```
int main(void) {
   int sum=0, a=1, b=1;

   while (sum < 20) {
      sum = a + b;
      a = b;
      b = sum;
      printf("%d\n", sum);
   }
}
```

Solution:

Sum	a	b	Printed on screen
0	1	1	
2	1	2	2
3	2	3	3
5	3	5	5
8	5	8	8
13	8	13	13
21	13	21	21

2 points for each correct line printed.
If the students have the correct table, but only the 21 printed, give them 10 points.

NAME:

2. (20 points total)

An experimental technique such as X-ray fluorescence measures the concentration of gas-phase species that passes through the beam from the source to the detector. Raj has run a simulation that generates the concentration of species along this exact beam length. His results are below:

Distance (cm)	Concentration (moles/liter)
0	10
1	15
2	13
3	7
4	9
5	11
6	18

To compare his results with the experimental results, Raj has to use numerical integration to compute the total absorbance of species, since

$$Absorbance = \int (Concentration)dx$$

a) Use the trapezoid rule and compute the integral from [0, 6]. (8 points)
b) Do not use the odd distances (1,3,and 5) and recompute the integral. (8 points)
c) How does the error in the Trapezoid rule scale with the number of data points used in computing the integral? (4 points)

a) Use formula: Integral = $h*[0.5*f_1 + f_2 + f_3 + \ldots + f_{n-1} + 0.5*f_n]$ (4 points)
h = 1, (2 points)
Integral = 1*[0.5*10+15+13+7+9+11+0.5*18] = 69 (2 points)

b) Use the same formula with:
h = 2, (4 points)
Integral = 2*[0.5*10+13+9+0.5*18] = 72 (4 points)

c) The error decreases as the number of data points increases (h decreases). (2 points). The error scales quadratically with h; error is proportional to h^2. (2 points)

NAME:

3. (25 points total)

Dave is in the process of writing a program to compute the grade distribution in the class. The only missing piece in his program is to compute the standard deviation of the class grades. Standard deviation is defined as

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n}(x_i - \bar{x})^2}$$

where σ is the standard deviation, x_i is the *i*th test score (*i* goes from 0…n), and $\bar{x}$ is the mean (average) test score in the class.

Write the function for Dave. The function should return the standard deviation of the test scores. He will use it to compute the standard deviation of your test scores!
(20 points out of 25)

The function prototype is

```
double std(double scores[], double mean, int num_scores)
{
  /* Your code here */
  double sigma, sum;
  int i;

  sum = 0;
  for (i=0; i<=num_scores; i=i+1) {
       sum = sum + (scores[i]-mean)*(scores[i]-mean);
  }
  sigma = sqrt((1.0/(num_scores-1))*sum);

  return sigma;
}

/* it is unclear from the problem if scores
goes from 0…num_scores or 0…num_scores+1, so
both will be accepted */
```

NAME:

(5 points out of 25)
please supply the missing line that calls std() in the following test function:

```
#include <stdio.h>

double std(double scores[], double mean, int num_scores);

int main(void){
  double grade[3]={80.,90.,100.};
  double sigma, average=90.;
  int Nscores=3;

  sigma = std(grade, average, Nscores);

  printf("the standard deviation is %f\n",sigma);
  return 0;
}
/* Nscores+1 will also be accepted */
(2 points for correct grade)
(2 points for correct mean)
(1 point for correct Nscores)
```

NAME:

4. (20 points)

Tim is having trouble debugging his function. He needs the loop to run through 3 times, but it doesn't do that. Help Tim fix his function by debugging the loop and fixing it so that it runs through 3 times. Then write out what the function will print out to the screen and what it will return when called.

```
int tim(void) {
   int count,b,c,d;
   int *p, *q;

   count = 0;
   b = 3;
   c = 2;
   d = 5;
   p = &c;
   q = &b;
   while (count<3) {

     if (b ==1) {
          printf("b equals one %d %d\n", c,d );
     }
     else {
          printf("b is not equal to one %d %d\n", c, d);
     }

     if (count%2==0) {
          *q = 1;
          c = d*b;
     }
     else {
          *q = 3;
          c = d*b;
     }

     count = count + 1; /* the bug to be fixed */
  }
  return c;
 }

     The code prints:
     b is not equal to one 2 5
     b equals one 5 5
     b is not equal to one 15 5

     The code returns 5
```

NAME:

b	c	d	p	q	count
3	2	5	c	b	0
1	5				1
3	15				2
1	5				3

5 points for finding bug.
3 points for each line printed correctly (and number returned) 2 point for each table row correct (if printing is wrong)

NAME:

5. (10 points)

Write a function prototype for a function named *comprate* with parameters of

- `N` (integer value of number of elements)
- `Temp` (array of doubles)
- `Avetemp` (the double value that this variable points to will be changed in the function)
- Function convert - a function with prototype:

```
        double convert(int N, double X)
```

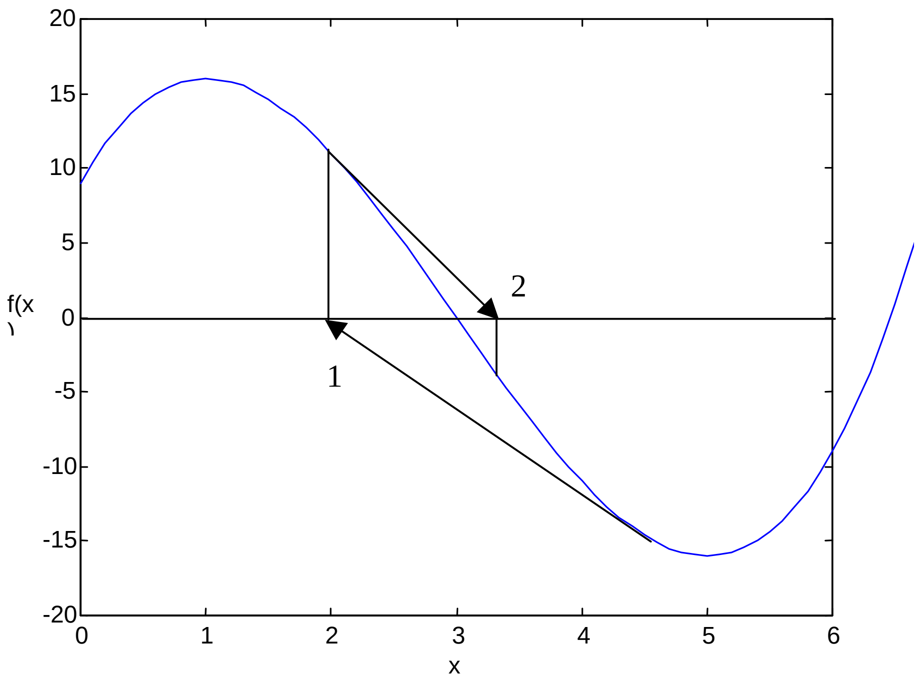
The function should return a double.

```
double comprate(int N, double Temp[], double *avetemp,
  double convert(int N, double X));
```

NAME:

6. (10 points)

You are given the following plot of a function, f(x).



If we want to find the root of f(x) at x=3, what range of initial guesses can we use for Newton's method? Draw out 2 iterations of Newton's method starting at x=4.5 (you do not need to use a ruler)

The range of initial guesses that can be made is very approximately between 1<x<5. (4/5 points). Points right around 1 and 5 may also not converge to the zero around 3, as the tangent may lead you away from the solution (tangents with a slope very close to 0). Mentioning this gives you an added point.