10.001 Introduction to Computer Methods  
Midterm Examination Fall Term 2002  
October 29, 2002  

Open book, notes and calculator  
Time allowed: 50 minutes  

Please write your name and username at the top of all pages before proceeding.  

There are 6 problems on this midterm. Please take a moment to read through the midterm in its entirety before beginning, and try not to spend too much time on any single problem.  

Answer all questions on the pages provided, and show all your work.
Problem #1 (15 points)

What, if anything, is wrong with the following fragments of code?

(a) int j,x[5]={0};
    for(j=1;j<=5;++j)
        x[j]=j;

    **accesses beyond end of array**

(b) void area(float r)
    {
        return 3.14159*r*r;
    }

    **incompatible variable types: float vs void**

(c) float a,*b;
    &a = b;

    **cannot assign an address**

(d) int j=32,k=32;
    j=pow(j,k);

    **$32^{32}$ exceeds the range of the int data type**
(f) double power( double x, int n) 
{
    if(n==1)
        return x;
    else
        return x*f(x);
    --n;
}

--n; is outside of if statement. This was intended to illustrate an infinite recursion. Due to a typographical error, power() and f() were inconsistent, making the answer unclear. Any reasonable answer was accepted for full credit.
Problem #2 (15 points)
Debug the following code, which has multiple errors (additional space between characters is provided for your convenience; you can insert the necessary corrections in the code provided, or rewrite the correct code below):

/* program to compute a raised to the –j power */
int main(void)
{

    int a, j,

    scanf("%d %d", a, j

    for( k=1; k=j ; k=k+1 )

        total = 1 / a * total

    printf("the answer is %f \n", total)

}

Declare total of type float and initialize to 1.
Declare k of type int
Add &'s to scanf()
Add semicolon to end of all lines
Add closing parenthesis on scanf()
Change k=j to a logical expression, k<j
Avoid integer division in calculating total (1.0 or explicit cast)
Problem #3 (10 points):

What gets printed by the following program?

```c
#include <stdio.h>
int func(int a)
{
    static int j=0;
    return(a*j++);
}
int main(void)
{
    int x=10,y=0;
    while(x>=0)
    {
        printf("%d %d\n",--x,func(x));
    }
    return (0);
}
```

```
9 0
8 8
7 14
6 18
5 20
4 20
3 18
2 14
1 8
0 0
-1 -10
```
Problem #4 (15 points)

(a) What is the order of the following polynomial?

\[x + 5x^2 + 8x^3 + 5x^5 = 0\]

fifth order

(b) Is the following equation linear or nonlinear?

\[xy + x \square y + 10 = 0\]

Linear in either x or y, whichever is considered variable
(This is ok for root-finding purposes. Subsequent to this midterm, we cover systems of linear equations, where there are multiple independent variables. In that case, both x and y may be variables. A second equation would be required to solve the system, and the xy term would make the system nonlinear.)

(c) Is the following equation algebraic or nonalgebraic?

\[1.5 \sin(x) + x^2 + 2x \square 1 = 0\]

algebraic

(d) Linearize the equation in part (c):

simple approach: drop sin and \(x^2\): \[2x - 1 = 0\]

better approach: write \(\sin(x)\) as Taylor series (see hw 3)
\[1.5x + 2x - 1 = 3.5x - 1 = 0\]
Question #5 (25 points)

The Numerical Recipes function to perform the bisection algorithm is `rtbis()`, shown below

```c
/* using bisection, find the root of a function func known to
 lie between x1 and x2. The root, returned as rtbis, will
 be refined until its accuracy is +/- xacc. */
#include <math.h>
#define JMAX 40

float rtbis(float (*func)(float), float x1, float x2, float xacc)
{
    void nrerror(char error_text[]);
    int j;
    float dx,f,fmid,xmid,rtb;

    f=(*func)(x1);
    fmid=(*func)(x2);
    if (f*fmid >= 0.0) nrerror("Root must be bracketed for
    bisection in rtbis");
    rtb = f < 0.0 ? (dx=x2-x1,x1) : (dx=x1-x2,x2);
    for (j=1;j<=JMAX;j++) {
        fmid=(*func)(xmid=rtb+(dx *= 0.5));
        if (fmid <= 0.0) rtb=xmid;
        if (fabs(dx) < xacc || fmid == 0.0) return rtb;
    }
    nrerror("Too many bisections in rtbis");
    return 0.0;
}
#undef JMAX
```

(a) Write the `prototype` for this function that should appear in the nr.h header file.

```
ffloat rtbis( float (*)(float), float, float, float);
```
(b) Complete the assignment statement for \( x_{acc} \) and fill in the invocation line in the following driver routine (you should not have to modify \( \text{rtbis}() \))

```c
#include <stdio.h>
#include "nr.h"
#include "nrutil.h"
float func(float);
int main(void)
{
    float xhi,xlo,relative_error,xacc,answer;
    scanf("%f %f %f", &xhi, &xlo, &relative_error)

    xacc = \text{relative_error}*\text{(xhi+xlo)}/2;

    /* insert invocation of bisection algorithm: */
    answer = \text{rtbis}( \text{func},xlo,xhi,xacc);

    \text{printf("the root is \%f\n",answer);
    return 0;
}
```

(c) For the equation shown below, write the function that would be called by \( \text{rtbis}() \) in order to locate its root.

\[ x^3 + e^x = \frac{1}{2} \]

```c
#include <math.h>
float func( float x)
{
    return x*x*x+exp(x)-0.5;
}
```

(d) What is the main advantage that bisection has over Newton’s Method for finding a root?

Once stared, bisection is guaranteed to converge on a root. Newton might not. Also, bisection does not require calculation of first derivative.
Problem #6 (20 points)
For the following questions, write a few lines of code that perform the tasks specified. You may assume that all variables you use have been properly declared, but not initialized. All code should conform to ANSI C standards.

(a) Write a C function that takes one parameter (n) and dynamically allocates the minimum amount of memory required for an array of integers (a vector) where the index runs from 1 to n. Show how this function would be called in a program (one line).

```c
int *dynamic_allocation(int n) {
    int *a;
    a = calloc(n,sizeof(int));
    return --a;
}
```

in main():
p = dynamic_allocation(n);

(could also use malloc() ).

(b) Write a fragment of code that reads values for an integer, a float and a double from a file called “input_file”, and writes them out to another file called “output_file”.

```c
FILE *ifp, *ofp;
ifp = fopen("input_file","r");
ofp=fopen("output_file","w");
fscanf(ifp,"%d%f%lf",&a,&f,&g);
fprintf(ofp,"%d %f %f\n",a,f,g);
...
fclose(ifp);
fclose(ofp);
```
(c) Write a fragment of code that does the same thing as part (b), but uses redirection to identify the input and output files. Show what the redirection command would look like (assume that the –o flag is NOT used when compiling this program.

```c
scanf("%d%f%lf", &a, &f, &g);
printf("%d %f %f\n", a, f, g);
```

with

```bash
a.out <input_file > output_file
```

(d) Write a fragment of code that works the same as the following for loop, but uses a while() construct:

```c
for(total=0; n>0; total=total+n*n, n=n-1);
```

```c
total=0;
while(n>0)
{
    total=total+n*n;
    n=n-1;
}
```