

Numerical Analysis Programming Assignments Submission Guidelines

The MS390 Numerical Analysis course will include some programming assignments, for which you will be asked to implement algorithms described in class or in the book. This document shows how to implement one such program, first by describing the algorithm as it will be given in class or in the book, then by implementing that algorithm in programs in each of four different formal programming languages: C, C++, Java, and Maple. When you write a program for an assignment, you may use any one of these four languages, but you *must* follow the guidelines given below (particularly as they pertain to input/output) in order to guarantee that the program you write on your computer, whatever operating system you use, will compile and run correctly on my computer, which may not use the same operating system (this is described below under the heading “Portability” for each language).

The example shown here implements the Bisection Method (see Section 2.1 in the book) for finding a root of a function of one variable. All four programs given below (in all four of the languages) compiled and ran, with output as shown, on my computer. If you have any doubts about how to implement something so that it will run as you wish on my computer, you should follow the models here. I will *not* modify your program to see if I can make it compile correctly on my computer — if it does not compile the first time on my computer, the program is *incorrect*.

Your code should be *readable* — in particular, you should follow common line-ending and indentation conventions for the language you are using. I encourage you to include comments in your programs, but they should be *relevant* comments.

When you submit a program, you must submit it by e-mail *before* the class in which the assignment is due. You may include your file either as a text-only file in an attachment or as text appended to the end of a text-only e-mail message. In either case, the “Subject:” line of the e-mail header should read “MS390 Programming Assignment.” If sending your file as an attachment, the file names

should follow the naming conventions described below under the heading “Naming” for each language (which, in particular, will specify the suffix on the name).

PSEUDOCODE

This is perhaps better described as “structured English” — it is *not* a structured programming language, but rather a general way of describing algorithms, indicating basic structures without specific reference to one particular programming language. Algorithms described in the book are described using Pseudocode, and those in class will also be described using Pseudocode.

Algorithm: Bisection Method, for approximating a single root x_r of the function $f(x)$ in the interval $[a, b]$ by finding c in $[a, b]$ so that $|c - x_r| < tol$.

```
f(x) = x2 - 2 (can use other functions)
Input a, b (endpoints)
Input tol (tolerance)
Input maxnum (maximum number of iterations)
if f(a) = 0 then Output a (already done)
else if f(b) = 0 then Output b
else if f(a) and f(b) have same sign
    then can't guarantee root in [a, b]
    — bad input error
else (ready for bisections)
    i := 0 (index to count iterations)
    while |b - a| > tol and f(a) ≠ 0
        and f(b) ≠ 0 and i < maxnum
        c := (a + b) / 2
        if f(a) and f(c) have same sign
            then a := c else b := c
        i := i + 1 (increment index)
    end while
    if i < maxnum Output c
    else exceeded maximum iterations — error
end if
end if
```

Notes:

- To check e.g. if $f(a)$ and $f(c)$ have same sign, check $f(a) \cdot f(c) > 0$ or something similar
- Terminal condition $|b - a| > tol$ works even if a and b are backwards

- Terminal conditions $f(a) \neq 0$ and $f(b) \neq 0$ guarantee proper termination even if happen on the root exactly at some point

C

This is a very common structured programming language, but can be a bit cryptic.

Program: bisect.c

```
#include <math.h>
#include <stdio.h>

double f(double x) {
    return x*x-2;
}

int main() {
    double a, b, c, tol;
    int maxnum;

    printf("First endpoint: ");
    scanf("%lf", &a);
    printf("Second endpoint: ");
    scanf("%lf", &b);
    printf("Tolerance: ");
    scanf("%lf", &tol);
    printf("Maximum iterations: ");
    scanf("%d", &maxnum);

    if (f(a)==0)
        printf("Answer: %f\n", a);
    else if (f(b)==0)
        printf("Answer: %f\n", b);
    else if (f(a)*f(b)>0)
        printf("Bad input\n");
    else {
        int i=0;
        while (fabs(b-a)>tol &&
              f(a)!=0 && f(b)!=0 && i<maxnum) {
            c=(a+b)/2;
            if (f(a)*f(c)>0) a=c;
            else b=c;
            i++;
        }
        if (i<maxnum)
            printf("Answer: %.10f\n", c);
        else {
            printf("Exceeded maximum iterations.\n");
            printf("left: %.10f; right: %.10f\n", a, b);
        }
    }
}
```

Output:

```
First endpoint: 1
Second endpoint: 2
Tolerance: 0.000001
```

Maximum iterations: 100000

Answer: 1.4142141342

Naming: File names for C source code files should take the form *filename.c*.

Portability: All input/output should use *only* the most basic formatted input/output procedures, `printf` and `scanf`.

For the curious: I will be compiling C programs using the GNU C compiler, `gcc` version 4.0.2, as bundled with Fedora Core Linux version 4. The `gcc` compiler is freely available under the GNU Public License.

C++

This is the programming language used in the CS231 Computer Programming I course.

Program: bisect.cpp

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

double f(double x) {
    return x*x-2;
}

int main(int argc, char ** args) {
    double a, b, c, tol;
    int maxnum;

    cout << "First endpoint: ";
    cin >> a;
    cout << "Second endpoint: ";
    cin >> b;
    cout << "Tolerance: ";
    cin >> tol;
    cout << "Maximum iterations: ";
    cin >> maxnum;

    cout << setprecision(10);
    if (f(a)==0)
        cout << "Answer: "
              << a << "\n";
    else if (f(b)==0)
        cout << "Answer: "
              << b << "\n";
    else if (f(a)*f(b)>0)
        cout << "Bad input\n";
    else {
        int i=0;
        while (fabs(b-a)>tol &&
              f(a)!=0 && f(b)!=0 && i<maxnum) {
            c=(a+b)/2;
            if (f(a)*f(c)>0) a=c;

```

```

        else b=c;
        i++;
    }
    if (i<maxnum)
        cout << "Answer: "
            << c << "\n";
    else
        cout
            << "Exceeded maximum iterations.\nleft: "
            << a << "right: " << b << "\n";
    }
    return 0;
}

```

Output:

```

First endpoint: 1
Second endpoint: 2
Tolerance: 0.000001
Maximum iterations: 100000
Answer: 1.414214134

```

Naming: File names for C++ source code files should take the form *filename.cpp*.

Portability: All input/output should use *only* the standard `cin` and `cout` formatted input/output stream objects associated with the `iostream` package, so your program should include the statement `#include <iostream>` at the beginning.

For the curious: I will be compiling C++ programs using the GNU C compiler, `gcc` (see above).

JAVA

This is the standard language for writing Web applets.

Program: `Bisect.java`

```

import java.io.*;

class Bisect {
    static BufferedReader ds =
        new BufferedReader(new
            InputStreamReader(System.in));

    private static double
    readDoubleFromKeybd() {
        try {
            return
                Double.valueOf(
                    ds.readLine()).doubleValue();
        }
        catch (IOException e) {return 0.0;}
    }

    private static int
    readIntegerFromKeybd() {
        try {
            return

```

```

                Integer.valueOf(
                    ds.readLine()).intValue();
        }
        catch (IOException e) {return 0;}
    }

    private static double f(double x) {
        return x*x-2;
    }

    public static void main(String[] args) {
        double a = 0.0;
        double b = 0.0;
        double c=0.0;
        double tol=0.0;
        int maxnum=100;

        System.out.print("First endpoint: ");
        a=readDoubleFromKeybd();
        System.out.print("Second endpoint: ");
        b=readDoubleFromKeybd();
        System.out.print("Tolerance: ");
        tol=readDoubleFromKeybd();
        System.out.print("Maximum iterations: ");
        maxnum=readIntegerFromKeybd();

        if (f(a)==0)
            System.out.print("Answer: "+a+"\n");
        else if (f(b)==0)
            System.out.print("Answer: "+b+"\n");
        else if (f(a)*f(b)>0)
            System.out.print("Bad input\n");
        else {
            int i=0;
            while (f(a)!=0 && f(b)!=0
                && Math.abs(b-a)>tol && i<maxnum) {
                c=(a+b)/2;
                if (f(a)*f(c)>0) a=c; else b=c;
                i++;
            }
            if (i<maxnum)
                System.out.print("Answer: "+c+"\n");
            else
                System.out.print(
                    "Exceeded maximum iterations.\n"+
                    "left: "+a+"; right: "+b+"\n");
        }
    }
}

```

Output:

```

First endpoint: 1
Second endpoint: 2
Tolerance: 0.000001
Maximum iterations: 100000
Answer: 1.4142141342163086

```

Naming: File names for Java source code files should take the form *filename.java*. As with any

Java application, the *filename* should match the name of the primary class in the application.

Portability: Portability should not be an issue with Java, since the language is designed with portability in mind. However, for overall consistency, all submitted Java programs should be stand-alone applications (not applets), and input/output should be handled using the methods of the objects `System.in` and `System.out`. Some care must be used with `System.in`, though, since it is designed to read in character strings only — the procedures `readDoubleFromKeybd` and `readIntegerFromKeybd` above perform the necessary conversions to `double` and `int` types.

For the curious: I will be compiling Java programs using the Java compiler included with the Java Developer Kit (JDK) 1.5 release distributed for Linux by Sun Microsystems.

MAPLE

This is not so much a *language* as a computer mathematics *system*, but it does include a fully functional structured programming language.

Program: Maple text file `bisect.txt`.

```
> f:=x->x^2-2:
> a:=1.0:
> b:=2.0:
> tol:=0.000001:
> maxnum:=100000:
> if f(a)=0 then a
> elif f(b)=0 then b
> elif f(a)*f(b)>0 then
> ERROR('Bad input', a, b):
> else
> i:=0:
> while abs(b-a)>tol and f(a)<>0
> and f(b)<>0 and i<maxnum do
> c:=(a+b)/2:
> if f(a)*f(c)>0 then a:=c else b:=c fi:
> i:=i+1:
```

```
> od:
> c
> fi;
```

Output:

```
1.414214136
```

Naming: Maple source code files should be generated in Maple using the menu command `File -> Export As`, and in the file selection window select the `Plain Text` type. As a result, file names will take the form *filename.txt*. (This is because the standard Maple worksheet format, given in files with names of the form *filename.mws*, is not easily human-readable.)

Portability: Portability should not be an issue with Maple, since it is a complete stand-alone system rather than a system-dependent compiler. However, the user should be aware of the limitations of Maple input/output — output is indicated by the use of a semicolon (“;”) rather than a colon (“:”) at the end of an execution group (note the use of the semicolon above), and input must be handled by directly coding the variable into the program (for readability, you should put such variables near the top of the program, as above). Also, your program should be a standard worksheet, not a Maplet (Maple applet).

Note: Maple will frequently have built-in procedures which perform the same task (perhaps using a different method) as that which you will be asked to program. You may compare the results of your program with these built-in procedures, but you must write your own procedure for the programming assignment.

For the curious: I will be running Maple programs using Maple 10 for Linux from Maplesoft, Inc. Maple is *not* free, but we do have a site license for a limited number of copies in the computing laboratories.