Instructions: (Read carefully before the exam begins.)

1. Before you begin, make certain that you have one exam booklet with pages numbered 1-20 printed double-sided.
3. The exam consists of 10 questions worth 10 marks each.
4. Write your answers with reasonable care. One mark per question may be deducted for readability.
5. Place all your answers in the space provided on these pages. Programming questions are to be answered in the C programming language.
6. Questions will not be interpreted during the exam. If you consider the wording of a question to be ambiguous, state your assumptions clearly and proceed to answer the question to the best of your ability. If you believe that there is an error in the exam, inform a proctor, who will investigate. If the exam does contain an error, an announcement will be made to the class as a whole.
7. Cheating is an academic offense. Your signature on this exam indicates that you agree to the University’s policies regarding cheating on exams.

Signature: ____________________________________ (in pen)
This space intentionally left blank.
Question 0

Write a program that prints your name.
Question 1

The following program prints exactly 10 lines. Write its output on the lines below it.

```c
#include <stdio.h>
void main() {
    int i = 2;
    i %= 3;
    printf("%d\n", i);
    int j = 23 / 2;
    printf("%d\n", j);
    i = j = 13;
    j /= 2;
    printf("%d\n", j);
    int k = ++i - j--;
    printf("i=%d\nj=%d\nk=%d\n", i, j, k);
    i = j = 6;
    do {
        --i;
        j++;
        printf("%d\n", i);
    } while( !( (i < 3) || (j > 8) ) );
    for(i = 0; i < 6; i++)
        printf("%d", i);
    printf("\n");
}
```

1
2
3
4
5
6
7
8
9
10
Question 2

The C math library contains the function `atan2()` that calculates the value of the arc tangent of $y/x$, using the signs of the two arguments to determine the quadrant of the result. It returns a value in the range $[\pi, -\pi]$.

Write a function to calculate the arc tangent using only the `atan()` function from the math library:

```c
double my_atan2(double y, double x);
```

`atan()` returns a value in the range $[\pi/2, -\pi/2]$. Use `assert()` to terminate the program if $x$ or $y$ equal 0.0 or -0.0. You may ignore cases where $x$ or $y$ equal $\pm\infty$ or `NAN`.

Sample test program

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

double my_atan2(double y, double x);

void main() {
    printf("%g\n", my_atan2(1, 1) * 180 / acos(-1));
    printf("%g\n", my_atan2(-1, -1) * 180 / acos(-1));
}
```

Sample output

```
45
-135
```

Question 3

Write a function

```c
void absDiff(double a[], double b[], double c[], int n);
```

that is given three arrays \( a, b, \) and \( c \) of the same size, \( n \geq 0 \). For each element \( i \) it places the absolute value of the difference between \( a[i] \) and \( b[i] \) in \( c[i] \).

*The next page is blank to provide extra space for your answer.*

Sample test program

```c
#include <stdio.h>

void absDiff(double a[], double b[], double c[], int n);

void main() {
    double d[3] = {0.0, -1e3, 6.0 };
    double e[3] = {-0.0, -1, 3.0 };
    double f[3];

    absDiff(d, e, f, sizeof(d)/sizeof(d[0]) );
    printf("%g %g %g\n", f[0], f[1], f[2]);
}
```

Sample output

```
0 999 3
```
You may use this page to answer question 3.
Question 4

Write a program that determines an election outcome. It first reads two integers from standard input: the first is the number of parties and the second is the number of ridings. It then reads the vote tallies from standard input, one integer per party per riding. The tallies are reported in the same order by party for each riding. There will be no ties within a riding. It stops reading vote tallies once it has enough data to determine the majority winner or it has read all of the vote tallies. The winner is reported by printing their place in the party order (i.e. 0, 1, or 2 if there are 3 parties). In the case of a tie, any one of the winners can be reported.

The next page is blank to provide extra space for your answer.

Sample Input

3 6
10 12 1
15 8 3
20 25 5
12 14 2

Sample Output

1
You may use this page to answer question 4.
**Question 5**

It stands to reason that the greatest common divisor of three numbers is the greatest common divisor of the one of the numbers and the greatest common divisor of the other two numbers. Write a function that returns the greatest common divisor of three numbers:

```c
int gcd(int a, int b, int c);
```

*The next page is blank to provide extra space for your answer.*

**Sample test program**

```c
#include <stdio.h>

int gcd(int a, int b, int c);

int main() {
    printf("%d\n", gcd(18, 12, 27));
    return 0;
}
```

**Sample output**

3
You may use this page to answer question 5.
Question 6

Write a program that reads a list of integers from standard input, one per line. When the program reads a non-positive integer, it stops. Otherwise, if the decimal representation of the integer contains a decimal digit not previously printed, it prints the integer.

The next page is blank to provide space for your answer.

Sample Input

99
100
57
88
87
13
44
45
7
62
301
0

Sample output

99
100
57
88
13
44
62
You may use this page to answer question 6.
**Question 7**

Show that \( f(x) = x \sin x \) has at least one real root in the interval \([1,10]\). \((3 \text{ marks})\)

Write a program to print one of the real roots of the function accurate to at least \(10^{-4}\), i.e. with absolute error \(\leq 10^{-4}\). Use the secant method. The secant method starts with guesses \(x_0\) and \(x_1\) and computes the next value of \(x\) by \(x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}\). Note that the secant method does not always converge to a root. \((7 \text{ marks})\)

*The next page is blank to provide extra space for your answer.*
You may use this page to answer question 7.
Question 8

The Sobel operator is used to detect edges in images. It takes a greyscale image in the form of an n x m array of greyscale values. Each array position represents a pixel in the image and the greyscale value is a number between 0 and 255, with 0 representing black, 255 representing white, and other values representing shades of grey between black and white (see left diagram). The Sobel operator uses two 3 x 3 filters, one of which is the horizontal filter in the right diagram.

\[
\begin{array}{ccc}
0 & 1 & 2 \\
128 & 128 & 64 \\
128 & 64 & 64 \\
64 & 64 & 0 \\
\vdots & \vdots & \vdots \\
\end{array}
\]

The center of the filter is moved over each pixel in the image (except the outside rows/columns) and the filter weights are multiplied by the underlying pixel values and summed to generate a filter value \( f \) for that pixel in the image. The absolute value of \( f \) or 255 is recorded, whichever is less. E.g. The filter value for pixel [1,1] in the above image is \((1 \times 128 + 0 \times 128 - 1 \times 64) + (2 \times 128 + 0 \times 64 - 2 \times 64) + (1 \times 64 + 0 \times 64 - 1 \times 0) = 256 \) which is recorded as 255. The filter values for pixels in the outside columns and rows is defined to be 0.

Write a function that applies a provided filter to an n x m image:

\[
void 
\begin{array}{l}
sobel(int image[], int filter[3][3], int filtered[], int n, 
\end{array}
\]

The next page is blank to provide extra space for your answer.

Sample Program

```c
#include <stdio.h>

void sobel(int image[], int filter[3][3], int filtered[], int n, int m);

void main() {
    int image[4][4] = {
        {1, 1, 1, 1}, {1, 1, 0, 0}, 
        {1, 1, 0, 0}, {1, 1, 1, 1} 
    };

    int filter[3][3] = {
        {1, 0, -1}, {2, 0, -2}, {1, 0, -1} 
    };

    int filtered[4][4];

    sobel((int*)image, filter, (int*)filtered, 4, 4);

    for(int i=0; i<4; i++) {
        for(int j=0; j<4; j++)
            printf("%3d ", filtered[i][j]);
        printf("\n");
    }
}
```

Sample Output

```
0   0   0   0
0   3   3   0
0   3   3   0
0   0   0   0
```
You may use this page to answer question 8.
Question 9

Situations may occur where it is required to manipulate integers that exceed the representation range of C integer types. In such cases, the digits of the integers could be stored in arrays. For example the decimal number 12345 could be represented by:

```
    int a[] = { 1, 2, 3, 4, 5 };
```

Implement a function

```
    void multiply(int a[], int b[], int c[], int n);
```

that takes three arrays `a`, `b`, `c`, all of size `n` and performs `c = a x b`. The decimal integers represented by `a` and `b` are unsigned. If the number of digits in the product `a x b` exceeds the size of `c`, use `assert()` to terminate the program with an error message.

The next page is blank to provide extra space for your answer.

Sample test program

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

void multiply(int a[], int b[], int c[], int n);

int main() {
    int a[4] = { 0, 0, 9, 9 };
    int b[4] = { 0, 1, 0, 0 };
    int c[4];
    multiply(a, b, c, 4);
    for(int i=0; i<4; i++) printf("%d", c[i]);
    return 0;
}
```

Sample output

```
9900
```
You may use this page to answer question 9.
This space intentionally left blank.