ECE 120 Worksheet 4: Problem Solving with C

As an engineer, you will often need to write programs to help you perform computations that cannot easily be done by hand. Typically, such calculations are expressed as complex mathematical formulae that need to be translated into a programming language. In this discussion, you will explore how various mathematical formulae can be translated to C constructs. As a reminder, there are only three basic programming constructs: sequential, conditional, and iterative.

As a reminder, the sum symbol $\sum$ directly translates into a for loop, e.g.,

$$s = 1 + 2 + \ldots + n = \sum_{i=1}^{n} i$$

becomes

```c
s = 0;
for (i = 1; i <= n; i = i + 1) {
    s = s + i;
}
```

where the variable $i$ is called the iteration variable.

Similarly, the product symbol $\prod$ directly translates into a for loop as well, e.g.,

$$p = 1 \times 2 \times \ldots \times n = \prod_{i=1}^{n} i$$

becomes

```c
p = 1;
for (i = 1; i <= n; i = i + 1) {
    p = p * i;
}
```
1. **Iterative Construct**

An iterative construct executes a statement (usually a compound statement) multiple times, until some condition becomes false.

1. Translate the following mathematical equation into C code using a `for` loop:

$$e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \cdots + \frac{x^n}{n!} = \sum_{i=0}^{n} \frac{x^i}{i!}$$

Assume that variables `n` and `x` have already been declared and assigned values.

```c
double sum = 0.0;
double term = 1.0;
int i;
for (i = 0; i <= n; i++) {
    sum = sum + term;
    term = term * (x / (i + 1));
}
```
2. Trace through your program for the case of $n = 4$ assuming that $x = 2$, and compare the result of another team’s program to the result you obtain while tracing your program. List which variables you are including in your trace and show how those variables change as you complete your trace.

<table>
<thead>
<tr>
<th>Iteration variable: $i$</th>
<th>Variable 1: $\text{term}$</th>
<th>Variable 2: $\text{sum}$</th>
<th>Variable 3: _______</th>
<th>Variable 4: _______</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.333</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.666</td>
<td>6.333</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.266</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Numerical Integration (not graded)

Here’s a thought problem for you. The code below calculates the integral of \( f(x) = x^2 \) by adding up area of \( N \) vertical slices. The area of each slice is calculated as the product of the width with the average height \( f(x) \) at the two sides of the slice. Mathematically, the approximation to the integral should approach the exact solution \( \left( \frac{1}{3} \right) \) asymptotically as \( N \) increases. The graph below shows the actual results as a function of \( N \). Can you explain what is happening? Can you come up with a way to fix the problem?

```c
#include <stdio.h>
#include <stdlib.h>

/* Calculate the integral of x^2 from 0 to 1, using N rectangles to estimate. */
int main ()
{
  unsigned N = 100000;
  float    sum;
  float    point;
  unsigned index;
  /* Start with 1/2 of the value at 0 and 1/2 of the * value at 1, scaled by 1/N. */
  sum = (0 * 0 + 1 * 1) / (N * (float)2.0);
  for (index = 1; N > index; index = index + 1) {
    /* Add each point, scaled by 1/N. */
    point = index / (float)N;
    sum   = sum + (point * point) / N;
  }
  printf ("With %u slices, the integral is \%f.\n", N, sum);
  return 0;
}
```

Integral result (sum) as a function of \( N \).