1 Problems

1. Write a function in SCHEME for computing the number of digits of a positive integer. You may assume the existence of the successor() function, which returns \((x + 1)\), when called with \(x\).

Solution: □

```
1: (define (numdigits n)
2:   (if (= (n div 10) 0) 1
3:     (successor (numdigits (n div 10))))
```

Algorithm 1.1: Computing the number of digits of a positive integer in Scheme.

2. Write a fragment in PROLOG that returns \(2^x\), when called with \(x\).

Solution: □

```
1: power2(U, 1) :- U = 0.
2: power2(U, V) :- not (U = 0), power2(U - 1, Y), V is 2 * Y.
```

Algorithm 1.2: Implementing the \(2^x\) function in Prolog.

3. As discussed in class, the C language permits only call-by-value as a parameter-passing mechanism. How then can the value of a variable be changed permanently within a function?

Solution: C permits you to pass the address of a variable to a function. Although the address is passed by its value, dereferencing the address gives the called function, the actual memory location to modify. For instance, consider the following block of code:

```
int main()
{
    int i;

    foo (int &i);
}

void foo (int *i)
{
    int *i=4;
}
```
In the above program the function 'foo' does modify the value of $i$ globally. 

4. Discuss how the following features have been promoted and violated in the C programming language: (a) Expressiveness, (b) Uniformity.

Solution:

(a) Expressiveness in C - The availability of recursion promotes expressiveness, while the lack of object-oriented features (such as classes) violates it.

(b) Uniformity in C - The fact that a semicolon can be used as a delimiter for statements and functions promotes uniformity, while the inability to overload the “+” operator to add arrays violates uniformity.

5. Assume that you are given a rudimentary programming language which contains only four operators, viz., $+,-, abs$ and $div$. $+$ and $-$ have their usual meanings, while $div(a,b)$ returns the quotient of $\frac{a}{b}$ and $abs(a)$ returns the absolute value of $a$. Write a C-style function $\text{max}(a,b)$ that takes two integers $a$ and $b$ as input and returns the maximum of the two. Note that you can only use the operators provided; in particular, the constructs "if", "while", and "for" are not available.

Solution: Let us study the function $f(a,b) = \text{div}(((a + b) + \text{abs}(b-a)),2)$. We consider the following three cases:

(i) $a > b$ - In this case $\text{abs}(b-a) = (a - b)$ and hence $f(a,b) = \text{div}(((a + b) + (a - b)),2) = \text{div}(2a,2) = a$.

(ii) $b > a$ - In this case $\text{abs}(b-a) = (b - a)$ and hence $f(a,b) = \text{div}(((a + b) + (b - a)),2) = \text{div}(2b,2) = b$.

(iii) $b = a$ - In this case $\text{abs}(b-a) = 0$ and hence $f(a,b) = \text{div}(((a + a)),2) = \text{div}(2a,2) = a$.

We see that in all three cases $f(a,b) = \text{max}(a,b)$, so we can indeed produce the maximum of two integers using only the operators provided! The formal algorithm is described below:

<table>
<thead>
<tr>
<th>Function</th>
<th>$\text{MAX}$ $(int\ a,\ int\ b)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$\text{return}(\text{div}(((a + b) + \text{abs}(b-a)),2))$</td>
</tr>
</tbody>
</table>

**Algorithm 1.3:** Implementing $\text{max}$ without $\text{if}$