Q1 - Recursion
What does the below function print for input recursion("Hello", 5)?
```cpp
void recursion(string s, int n) {
    if (n == 0) return;
    cout << s << ";
    recursion(s, n - 1);
}
```

Q2 - Pass by Value / Reference
What does the below program print?
```cpp
void timesTwo(int x) {
    x = 2*x;
}
int main() {
    int n = 42;
timesTwo(n);
cout << n << endl;
}
```

Q3 - Division
What is the expected output of below:
- a) 4 + 7 / 3.0
- b) 5 / 2

Q4 - Conditionals
```cpp
    cin >> response;
    if (response == 'H') {
        cout << "Hello, world!" << endl;
    } else {
        cout << "Goodbye, world!" << endl;
    }
```
Provided the above code snippet, what will be printed with the following user inputs?
1. H
2. G
3. g
4. h

Q5 - Loops on Arrays
What is the output from the below code snippet?
```cpp
int numbers[] = {1, 2, 3};
for (int i = 0; i < 4; i++) {
    cout << numbers[i] << endl;
}```
CSCI 262 Lecture 3 – Pointers

Outline

• Numbers & numbering systems: positional notation, base 2 and other bases used by computer scientists, bits & bytes

• Pointers & memory: addresses, values, address-of operator, pointers defined, dereference operator, pointers to pointers, pointer variables, nullptr and NULL

• Pointer notes: different ways of dereferencing, declaration syntax

Readings

On pointers: section 7.1 in your textbook

On numbering systems: Appendix F in your textbook

History of the place-value system (positional notation) in the West:

• https://en.wikipedia.org/wiki/History_of_the_Hindu%E2%80%93Arabic_numeral_system

Self Check

1. A 32-bit unsigned int can store non-negative integers up to a maximum of __________.

2. The value stored at a particular address in memory depends on the __________ of the value.

3. Using the operator __________, we can obtain the address of the value stored in a variable. The address can be stored in a __________ variable.

4. Using the operator __________, known as the __________ operator, we can get or set the value pointed to by a pointer.

5. The __________ keyword stands for a pointer that does not point to anything; this is a good value to use when initializing pointer variables.

For Further Practice

What are the values of x and y after executing the following code?

```c
int x = 0, y = 0;
int *p = &x;
int *q = &y;
*p = 42;
p = q;
*p = 77;
q = nullptr;
```