Programming Questions

Questions and Solutions

1. **Write a recursive function to calculate the sum of digits of a number**

```c
int sumOfDigits(int n) {
    if (n == 0)
        return 0;
    return (n % 10) + sumOfDigits(n / 10);
}
```

2. **Write a function to rotate an array by k positions using pointers**

```c
void rotateArray(int *arr, int size, int k) {
    k = k % size; // Normalize k
    int temp[k];

    // Store first k elements
    for(int i = 0; i < k; i++)
        temp[i] = arr[i];

    // Shift remaining elements
    for(int i = k; i < size; i++)
        arr[i-k] = arr[i];

    // Place temp elements at end
    for(int i = 0; i < k; i++)
        arr[size-k+i] = temp[i];
}
```

3. **Write a recursive function to find GCD of two numbers**

```c
int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}
```

4. **Write a function that returns a pointer to a dynamically allocated array of prime numbers up to n**

```
int* generatePrimes(int n, int* count) {
    int* primes = (int*)malloc(n * sizeof(int));
    *count = 0;

    for(int i = 2; i <= n; i++) {
        int isPrime = 1;
        for(int j = 2; j * j <= i; j++) {
            if(i % j == 0) {
                isPrime = 0;
                break;
            }
        }
        if(isPrime)
            primes[(*count)++] = i;
    }
    return primes;
}
```

5. **Write a recursive function to reverse a string**

```
void reverseString(char* str, int start, int end) {
    if(start >= end)
        return;

    char temp = str[start];
    str[start] = str[end];
    str[end] = temp;

    reverseString(str, start + 1, end - 1);
}
```

6. **Write a function that takes an array of pointers to strings and sorts them alphabetically**

```
void sortStrings(char** strings, int n) {
    for(int i = 0; i < n-1; i++) {
        for(int j = 0; j < n-i-1; j++) {
            if(strcmp(strings[j], strings[j+1]) > 0) {
                char* temp = strings[j];
                strings[j] = strings[j+1];
                strings[j+1] = temp;
            }
        }
    }
}
```

7. **Write a recursive function to check if an array is sorted**

```c
int isSorted(int* arr, int size) {
    if(size <= 1)
        return 1;
    if(arr[size-1] < arr[size-2])
        return 0;
    return isSorted(arr, size-1);
}
```

8. **Write a function that uses pointers to merge two sorted arrays**

```c
int* mergeSortedArrays(int* arr1, int size1, int* arr2, int size2) {
    int* result = (int*)malloc((size1 + size2) * sizeof(int));
    int i = 0, j = 0, k = 0;

    while(i < size1 && j < size2) {
        if(arr1[i] <= arr2[j])
            result[k++] = arr1[i++];
        else
            result[k++] = arr2[j++];
    }

    while(i < size1)
        result[k++] = arr1[i++];
    while(j < size2)
        result[k++] = arr2[j++];

    return result;
}
```

9. **Write a recursive function to generate all binary strings of length n**

```c
void generateBinary(char* str, int n, int i) {
    if(i == n) {
        str[i] = '\0';
        printf("%s\n", str);
        return;
    }

    str[i] = '0';
    generateBinary(str, n, i + 1);
    str[i] = '1';
    generateBinary(str, n, i + 1);
}
```

10. **Write a function that implements matrix multiplication using pointers**

```
void matrixMultiply(int* A, int* B, int* C, int rows1, int cols1, in
    for(int i = 0; i < rows1; i++) {
        for(int j = 0; j < cols2; j++) {
            *(C + i*cols2 + j) = 0;
            for(int k = 0; k < cols1; k++) {
                *(C + i*cols2 + j) += *(A + i*cols1 + k) * *(B + k*c
            }
        }
    }
}
```