Midterm cs280. Digipen login:_____

1. Given an array of $n = 2^m$ elements, write a **recursive** function `int product(int* array, int begin, int end)` that takes an array and first (`begin`), one past the last (`end`) indices, and returns a **product** of all elements in the range. The function **should split the array into 2 equally sized parts**.

   ```
   int product(int* array, int begin, int end) {
   ```

2. True/false

   (a) $\log n \in O(n^3)$

   (b) $\log n \in O(\sqrt{n})$

   (c) $2^n \in O(n^2)$

   (d) $2^n \in O(3^n)$

   (e) $3^n \in O(2^n)$

3. Given an **unsorted** array of $n = 2^m$ elements, write a **recursive** function `int find(int* array, int begin, int end, int value)` that takes an array and first (`begin`), one past the last (`end`) indices, and returns an index of an element that is equal to `value` (assume there is only one such element). If element with the specified value does not exist – return `begin-1` . The function **should split the array into 2 equally sized parts**.

   ```
   int find(int* array, int begin, int end, int value) {
   ```

4. You have an executable `alg.exe` which implements an unknown algorithm. `alg.exe` works on a sequence of numbers – say, provided as command line arguments:

   ```
   alg.exe 12 4 7 3 31 1
   ```

   You know that run-time complexity is either $O(n^2)$ or $O(n^3)$, where n is the length of the (command line) sequence.

   Describe your steps to experimentally determine algorithm's run-time complexity.

5. Write C++ function that inserts a given value at the front of a singly linked list. Assume the following `Node` class and `main` below. No global variables.

```
struct Node { // do not modify
    int value;
    Node *next;
};

_____ push_front ( ____________ ) {









}
int main() {
    Node * pList = NULL;       /* create empty list */
    push_front( 5 __________ );    /* insert 5 into pList, note that
                                      returned value of the function is not used */
    push_front( 7 __________ );    /* insert 7 into pList */
}
```

6. Evaluate the following postfix notation arithmetic expression:
   `174-+42+*23*-`.
   All arguments are single-digit.

7. Convert the following infix notation arithmetic expression to postfix:
   `(5-4)*(6+3+2+9)-8*7`

8. What is the runtime of each of the following algorithms that combine 2 sorted array into a single sorted array. Assume both input arrays are size $n/2$, no duplicates – all n values are different, output array is already allocated and has size n.

   What is the worst case runtime of each of the algorithms, what is the best runtime of each of the algorithms? Make sure to specify the conditions that lead to each of the cases(best and worst).

```
Alg1( int * a, int * b, int * out ) {
    for ( int i = 0; i < n/2; ++i ) { out[i] = a[i]; }

    for ( int j = 0; j < n/2; ++j ) {
        out[n/2 + j] = b[j];
        int k = n/2 + j; // position of just inserted element
        while ( out[k-1] > out[k] && k>0 ) {
            std::swap( out[k-1], out[k] );
            --k;
        }
    }
}

Alg2( int * a, int * b, int * out ) {
    int i = 0, j = 0, k = 0;
    while ( i < n/2 && j < n/2 ) {
        if ( a[i] < b[j] ) {
            out[k] = a[i];
            ++k; ++i;
        } else {
            out[k] = b[j];
            ++k; ++j;
        }
    }

    if ( i < n/2 ) {
        while ( i < n/2 ) {
            out[k + i] = a[i];
            ++i;
        }
    } else {
        while ( j < n/2 ) {
            out[k + j] = b[j];
            ++j;
        }
    }
}
```

9. What is the runtime and memory requirements for each of the following algorithms that reverse an array? When calculating memory requirements include stack (actually there are no dynamic memory allocation in any of the algorithms). Assume int and pointer are 4 bytes. Be specific, show intermidiate steps. Answer `3n` with no explanation will receive 0 points.

   - Runtime of rec1

   - Memory requirement of rec1

   - Runtime of rec2

   - Memory requirement of rec2

```
void rec1( int* a, int n )
{
    if ( n < 2 ) return;
    swap( a[0], a[n-1] );
    rec1( a+1, n-2 );
}

void rec2( int* a, int n, int d )
{
    if ( d == n ) {
        return;
    }
    int t = a[d];
    rec2( a, n, d+1 );
    a[n-1-d] = t;
}
```

10. Rewrite recursive function below to take advantage of tail recursion optimization. The new function has to be recursive, you may modify the signature.

```
long sum( int n ) {
    if ( n == 0 ) return 0;
    return n + sum(n-1);
}
```