William Kelley
Assignment 4 - CS472

Number 1

```
// Source: https://www.geeksforgeeks.org/coin-change-dp-7/

#include <iostream>
#include <iomanip>
#include <string>
#include <array>
#include <fstream>

using namespace std;

int count( int S[], int size, int n )
{
    // If n is 0 then there is 1 solution
    // (do not include any coin)
    if (n == 0)
        return 1;

    // If n is less than 0 then no
    // solution exists
    if (n < 0)
        return 0;

    // If there are no coins and n
    // is greater than 0, then no
    // solution exist
    if (size <=0 && n >= 1)
        return 0;

    // count is sum of solutions (i)
    // including S[size-1] (ii) excluding S[size-1]
    return count( S, size - 1, n ) + count( S, size, n-S[size-1] );
}

int main()
{
  const int SIZE = 4;
  int num;
  int nums[SIZE] = {1, 3, 5, 9};

  cout << "Enter number desired to search number of solutions of(0 to
exit): ";
  while(cin >> num && num != 0){
    cout << count(nums, SIZE, num) << endl;
```

```
    cout << "Enter number desired to search number of solutions of(0
to exit): ";
  }

  return 0;
}

/*
Williams-MacBook-Pro:Assignment 04 williamkelley$ ./coins
Enter number desired to search number of solutions of(0 to exit): 1
1
Enter number desired to search number of solutions of(0 to exit): 2
1
Enter number desired to search number of solutions of(0 to exit): 3
2
Enter number desired to search number of solutions of(0 to exit): 4
2
Enter number desired to search number of solutions of(0 to exit): 5
3
Enter number desired to search number of solutions of(0 to exit): 6
4
Enter number desired to search number of solutions of(0 to exit): 7
4
Enter number desired to search number of solutions of(0 to exit): 8
5
Enter number desired to search number of solutions of(0 to exit): 0
*/
```

Number 2

```
// Source: https://stackoverflow.com/questions/31865203/algorithm-
parenthesize-of-string-from-dynamic-programming-by-vazirani-et-al

#include <iostream>
#include <iomanip>
#include <math.h>

using namespace std;

char alphabet[] = "abc";
char multiplicationTable[3][3] = {
  { 'b', 'b', 'a' },
  { 'c', 'b', 'a' },
  { 'a', 'c', 'c' }
};  // Dimensions are k*k.

char *s = "ccccaa";

int N = strlen(s);
```

```
int k = strlen(alphabet);

/* Recursive function that returns 1 if it is
 * possible to get symbol from
 * string s of length n.
 */
int isSymbolPossible(char *s, char symbol, int n) {
    int i, j1, j2;
    if (n == 1) {
        return *s == symbol;
    }
    /* Loop over all possible ways to split the string in two. */
    for (i=0; i < n - 1; i++) {
        /* For each such subdivision, find all the multiplications
that yield the desired symbol */
        for (j1 = 0; j1 < k; j1++) {
            for (j2=0; j2 < k; j2++) {
                if (multiplicationTable[j1][j2] == symbol) {
                    /* Check if it is possible to get the required
left and right symbols for this multiplication */
                    if (isSymbolPossible(s, alphabet[j1], i+1) &&
                            isSymbolPossible(s+i+1, alphabet[j2], n -
i - 1)) {
                        return 1;
                    }
                }
            }
        }
    }
    return 0;
}

int main() {
    if (isSymbolPossible(s, 'a', N) == 1){
        cout << s << endl;
        cout << "Yes!\n";
    } else {
        cout << s << endl;
        cout << "No...\n";
    }
    return 0;
}

// acaca
// Yes!
// bbbba
// Yes!
// cccc
// No...
```

```
// ccccaa
// No...
```

Number 3

```
--] Source: https://codegolf.stackexchange.com/questions/49050/tiling-
a-2n-by-2n-grid-with-l-shaped-trominoes

c=cycle
z o(#)(x,y)=zipWith o(1#x)(2#y)
f n x y=unlines$(z(+)(\m w->[c[0,m]!!div(w-1)(2^(n-k))|k<-[1..n]])
(x,y),"O")%n
(_,x)%0=[x]
((p: o), x) % k=z(+ +)(\_ q -> ((o, x): c[(c[3 - q], [" |-+| +--+ |
+-|"!!(4 * p + q)])])!!abs(p - q) % (k - 1)) = << [(0, 1), (2, 3)]

--]   putStr $ f 4 5 6

--] +--++--++--++--+
--] |+-||-+||+-||-+|
--] ||+--+||||+--+||
--] +-|+-|-++-|-+|-+
--] +-||-+-++--+||-+
--] ||+-O||||-+|-+||
--] |+-||-+|-+|||-+|
--] +--++--+||-++--+
--] +--++-|-+|-++--+
--] |+-|||+--+|||-+|
--] ||+-|+-||-+|-+||
--] +-||+--++--+||-+
--] +-|+-|-++-|-+|-+
--] ||+--+||||+--+||
--] |+-||-+||+-||-+|
--] +--++--++--++--+
--]    putStr $ f 3 2 2

--] +--++--+
--] |O|||-+|
--] |-+|-+||
--] +--+||-+
--] +-|-+|-+
--] ||+--+||
-- ]|+-||-+|
--] +--++--+
--]    putStr $ f 2 1 1

--] O|-+
--] -+||
--] |-+|
```