CSCI 3333 Practice Midterm #4

- Do not start until instructed to do so.
- Write your UTRGV ID only in the space provided at the top of this page.
- The midterm is closed - no books, notes, computers, cell phones, calculators, etc.
- The time allotted for the exam is 70 minutes.
- There are 7 questions worth 28 points total; each problem is worth 4 points.
- These are not the actual midterm questions.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>
**Problem 1.** Complete the following implementation of a function `subset_sum` that returns whether some subset of an input set \( S \) sums to \( w \) using brute force.

```cpp
bool subset_sum(set<int> &partial, set<int>::iterator it, 
                set<int> &S, int w)
{
    if (it == S._____())
    {
        int tot = 0;
        for (int x : partial)
            tot += x;

        return (tot _____ w);
    }

    int cur = *it;
    ++it;
    if (subset_sum(partial, it, S, w))
        return _____;

    partial.insert(______);
    if (subset_sum(partial, it, S, w))
        return _____;

    partial.erase(______);
    return false;
}

int main()
{
    set<int> S{1, 4, 5, 8, 11, 16, 17, 22, 23};
    set<int> partial;
    cout << "The set {1, 4, 5, 8, 11, 16, 17, 22, 23} sums to 30: ";
    cout << subset_sum(partial, S.begin(), S, 30) << endl;
}
**Problem 2.** Fill in the table in Figure 1 of the minimum number of coins from the set \( S = \{1, 3, 7\} \) that sum to the given amount.

![Figure 1: The table for the first part of Problem 2.](image)

Fill in the table in Figure 2 of the minimum number of coins from the set \( S = \{1, 4, 5\} \) that sum to the given amount.

![Figure 2: The table for the second part of Problem 2.](image)

**Problem 3.** Let \( \text{DTIME}(f(n)) \) be the set of problems that have \( O(f(n)) \)-time worst-case algorithms. Fill in the blanks in the Venn diagram in Figure 3, which assumes \( P \neq \text{NP} \).

![Figure 3: The Venn diagram for Problem 3.](image)
Problem 4. Complete the following function make_change that decides whether change can be made for \( w \) using coins from a set of denominations \( C \) using dynamic programming.

```cpp
bool make_change(set<int> C, int w)
{
    bool* T = bool[w+1];
    for (int i = 0; i < w+1; ++i)
        T[i] = _____;
    for (int c : C)
        T[c] = _____;
    for (int sub_w = 1; sub_w <= w; ++sub_w)
        for (int c : C)
            if (sub_w-c > 0 && T[sub_w-c] == _____)
                T[sub_w] = _____;
    bool soln = T[_____];
    delete[] T;
    return soln;
}
```

```cpp
int main()
{
    set<int> S{1, 4, 5, 9};
    cout << "Change for 30 can be made using coin values 1, 4, 5, 9: ";
    cout << make_change(C, 30) << endl;
}
```
Problem 5. Identify what is known about each of the following problems. For each problem, check \textit{all} boxes that are true (there may be multiple).

Input: a graph $G = (V, E)$ and $k \in \mathbb{N}$.
Output: whether exist $v_i, v_j \in V$ with $d(v_i, v_j) > k$.
\(\square \text{ In P} \quad \square \text{ In NP} \quad \square \text{ NP-complete}\)

Input: a composite number $x \in \mathbb{N}$.
Output: a factor of $x$.
\(\square \text{ In P} \quad \square \text{ In NP} \quad \square \text{ NP-complete}\)

Input: an undirected graph $G$.
Output: whether $G$ has a cycle of length 5.
\(\square \text{ In P} \quad \square \text{ In NP} \quad \square \text{ NP-complete}\)

Input: an undirected graph $G = (V, E)$.
Output: whether $G$ has a cycle of length $|V|$.
\(\square \text{ In P} \quad \square \text{ In NP} \quad \square \text{ NP-complete}\)

Problem 6. Fill in the blanks:

When reducing 4SAT to 3SAT, replace each clause \((x_1 \lor x_2 \lor x_3 \lor x_4)\) with two clauses \((x_1 \lor x_2 \lor y)\) and \((\_ \lor x_3 \lor \_)\).

When reducing NAESAT to 3SAT, replace each clause \((x_1 \lor x_2 \lor x_3)\) with two clauses \((x_1 \lor x_2 \lor x_3)\) and \((\_ \lor \_ \lor \_ )\).

When reducing 2SAT to 3SAT, replace each clause \((x_1 \lor x_2)\) with a clause \((x_1 \lor x_2 \lor \_)\).

When reducing 1-in-3-SAT to 3SAT, replace each clause \((x_1 \lor x_2 \lor x_3)\) with four clauses \((x_1 \lor x_2 \lor x_3), (\neg x_1 \lor \neg x_2 \lor F), (\neg \_ \lor \_ \lor F), (\_ \lor \_ \lor F)\).
**Problem 7.** Complete the following implementation of a function `tribonacci` that computes elements of the *Tribonacci sequence* using memoization.

```cpp
int tribonacci(int n, map<int, int> &T)
{
    if (T.find(n) != T.end())
        return T[n];

    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    if (n == 2)
        return 1;

    int tn = tribonacci(n - 3, T) + tribonacci(n - 2, T) + tribonacci(n - 1, T);
    T[n] = tn;
    return T[n];
}

int main()
{
    cout << "The 10th Tribonacci number is ";
    map<int, int> T;
    cout << tribonacci(10, T) << endl;
}
```