Problem 1. True/False Questions (20 pts - 2pts each)
Write down T (True) or F (False) for each of the statement below.

1. ( ) In C++, `float` data type uses 4 bytes (32bits) and `double` uses 8 bytes (64bits) for most widely used platforms.
2. ( ) The number of values that `double` data type can represent is infinite.
3. ( ) The worst-case time complexity of binary search is $\Theta(\log n)$, assuming all elements are unique.
4. ( ) If all the elements in an array are single-digit integers, it is possible to sort the array in linear time complexity.
5. ( ) Quicksort has better worst-case time complexity than the MergeSort does.
6. ( ) When search is much more frequent than insertions or deletions in a container, `SortedArray` is a more efficient container than `List`.
7. ( ) The chance of key collision is equal or smaller in a chained hash than an open hash given a number of possible hash values (slots).
8. ( ) Dijkstra’s algorithm is not guaranteed to work correctly if there are negatively weighted edges.
9. ( ) The fastest algorithm of matrix multiplication has $\Theta(n^2)$ time complexity.
10. ( ) If $u$ and $u$ are $n \times 1$ vectors, and $A$ and $B$ are $n \times n$ matrices, $((u^T A) B) v$ has better time complexity than $((u^T (AB)) v)$.

Problem 2. Short answer questions (20pts - 4pts each)
Write down the expected output for each program (4pts each).

(a)
```cpp
#include <iostream>
int main() {
    std::cout << ( 1 + 2 * 3 ) << std::endl;
    return 0;
}
```

(b)
```cpp
#include <iostream>
int main() {
    int a[3] = {1,2,3};
    int* b = &a[1];
    b[1] = 10;
    std::cout << a[0] << "," << a[1] << "," << a[2] << std::endl;
    return 0;
}
```
Problem 3 - Long answer questions (30pts - 10pts each)

(a) Fill in the body of the `cumsum()` function so that the following properties hold upon the completion of the function.

\[
\text{dst}[i] = \sum_{j=0}^{i} \text{src}[j]
\]

for \(i = 0, \ldots, n - 1\). Note that the function must have \(\Theta(n)\) time complexity.
// function to calculate cumulative sums  
// INPUT:  
// n : number of elements in the arrays  
// src is size-n array, containing integer values  
// OUTPUT:  
// dst is size-n array, uninitialized  
// when the function ends, dst should contain  
// dst[i] = \sum_{j=0}^{i} src[j]  
void cumsum(int n, int* src, int* dst) {  
  // fill the blank below

(b) Complete the three lines in the search() function of MySortedArray.

  // "std::vector<T> data" is assumed to be declared above as a member variable  
  // search() function returns the index of the element x,  
  // -1 is returned when the element x does not exist  
  template <class T>  
  int mySortedArray<T>::search(const T& x) {  
      return search(x, 0, size-1);  
  }  
  template <class T>  
  int mySortedArray<T>::search(const T& x, int begin, int end) {  
      if ( begin > end )  
          return -1;  
      else {  
          int mid = (begin+end)/2;  
          if ( data[mid] == x )  
              return // **** FILL IN LINE 1  
          else if ( data[mid] < x )  
              return // **** FILL IN LINE 2  
          else  
              return // **** FILL IN LINE 3  
      }  
  }  

(c) Write down the expected output of the following code.

#include <iostream>  
void towerOfHanoi(int n, int s, int i, int d) {  
  std::cout << "towerOfHanoi called at n = " << n << std::endl;  
  if ( n > 0 ) {  
      towerOfHanoi(n-1,s,d,i);  
  }  
  towerOfHanoi(n-1,i,d,s);  
}  

// towerOfHanoi(4, 1, 3, 2)
Problem 4 - Dynamic Progsamming (20pts - 5pts each)

Consider the following Manhattan tourist problem. The weight at each edge represents the time cost moving from one node to the other. Let \((i, j)\) represents the node at row \(i\), column \(j\).

(a) How many possible paths do exist from node \((1, 1)\) (top-left) to node \((4, 3)\) (bottom-right)??

(b) Suppose that the following algorithm is used.

(a) If current node is \((i, j)\), look at the next possible paths and choose the path with smaller weight
(b) Repeat the above procedure until it reaches the destination

What is the overall cost from \((1, 1)\) to \((4, 3)\) if traversed based on the algorithm above? Write the cost from the source to each node along the path.
(c) Let \( d(i, j) \) be the optimal cost to reach to the node \((i, j)\). Write down a dynamic programming formulation to represent \( d(i, j) \) as a function of optimal costs in the preceding nodes.

(d) Fill all the node with \( d(i, j) \) in the figure below and highlight the path providing the optimal cost.
Problem 5. Hidden Markov Model (10pts)

Consider a $n$-state hidden Markov model $\lambda = \{\pi, A, B\}$ across $t$ time points where $\pi$ is prior distribution $A$ and $B$ represent transition and emission probability. (Same model as described in the class). Let $q_1, \cdots, q_T$ be the hidden state at time $1, \cdots, T$ and $o_1, \cdots, o_T$ be the observed outcomes.

(a) Given $q_1, \cdots, q_T$, are $o_1, \cdots, o_T$ conditionally independent of each other? Describe why briefly.

(b) Given $o_1, \cdots, o_T$, are $q_1, \cdots, q_T$ conditionally independent of each other? Describe why briefly.