

Winter 2019 Bridge Practice Exam 2

Take this exam under exam conditions. The questions you see may be harder or easier than what you'll see on the actual exam.

1. Induction proof (short answer)

- Prove that $n! > 2^n$ for $n \geq 4$

Base Case: let $n = 4$ such that $4! \geq 2^4 \rightarrow 24 \geq 16$

Inductive Step: Suppose that for some integer k , we have that $k! \geq 2^k$. Now we want to prove that $(k + 1)! \geq 2^{k+1}$.

$$(k + 1)! \geq (k + 1) \cdot k! \geq (k + 1) \cdot 2^k$$

Now, we know that this step that $k + 1 \geq 2$ for all $k \geq 4$

So we have that $(k + 1) \cdot 2^k \geq 2 \cdot 2^k = 2^{k+1}$

Therefore, by induction, we have that $n! \geq 2^n$ for $n \geq 4$

2. Probability Question (MC)

- Let A and B be two events. Suppose that the probability that neither A or B occurs is $2/3$. What is the probability that one or both occur?

(a) $2/3$

(b) $1/2$

(c) $1/3$

(d) $1/4$

Explanation: If $P((A \cup B)^C) = 2/3$, the complementary event is $P(A \cup B)$ which would then be $1/3$ because $P((A \cup B)^C) + P(A \cup B) = 1$

3. Counting Question (MC)

- There are 6 men and 7 women in a ballroom dancing class. If 4 men and 4 women are chosen and paired off, how many pairings are possible?

(a) $4!$

(b) $P(6, 4) * P(7, 4) * 4!$

(c) $C(6, 4) * C(7, 4) * 4!$

(d) $C(6, 4) + C(7, 4) + 4!$

Explanation: You must choose 4 men from 6 which is $C(6, 4)$. Then you have to choose 4 women from 7 which is $C(7, 4)$. Finally, you have to find the number of possible pairings for 4 couples. This calculation is $4!$. Therefore, the answer is $C(6, 4) * C(7, 4) * 4!$

4. Probability Question (Short answer)

- Suppose that $P(A) = 0.4$, $P(B) = 0.3$, and $P((A \cup B)^C) = 0.42$. Are A and B independent? Why?

Two events A and B are independent if $P(A \cap B) = P(A) \cdot P(B)$

Notice that we have $P(A \cup B) = 1 - P((A \cup B)^C) = 1 - 0.42 = 0.58$

By inclusion-exclusion principle, we have that $P(A \cup B) = P(A) + P(B) - P(A \cap B)$. Plugging in the number we have, we get that $P(A \cap B) = 0.4 + 0.3 - 0.58 = 0.12$

Since we have that $P(A \cap B) = P(A) \cdot P(B)$, we know that A and B are independent.

5. Pointer Question (MC)

- Determine the output of the following code:

(a) 20

(b) 25

(c) 30

(d) 35

```
// For question 5
int f(int* n, int m) {
    *n = 10;
    m = 10;
    return *n + m;
}

int main() {
    int n = 5;
    int m = 5;
    int res = f(&n, m);
    std::cout << res + n + m << std::endl;
}
```

Explanation: `n` is changed to 10 in the function `f`. Then `m` remains unchanged. The result of `f(&n, m) = 20`. Summing all that up, we get 35.

6. Counting Question (Short answer)

- How many arrangements are there of the word PROBABILITY?

The word PROBABILITY has 11 words. There are $11!$ arrangements not accounting for duplicates, but we have to account for them. Notice that B occurs twice and I occurs twice. Therefore, our solution is $\frac{11!}{2!2!}$

7. Algorithm Analysis (MC)

- What is the worst-case runtime of the following algorithm?

(a) $O(n^2)$

(b) $O(n * m)$

(c) $O(m^2)$

(d) $O(n + m)$

C++

```
// For question 7
void f(int* n, int* m, int n_size, int m_size) {
    for (int i = 0; i < n_size; i++) {
        for (int j = 0; j < m_size; j++) {
            // Some O(1) operation here
        }
    }
}
```

8. Algorithm Analysis (MC)

- What is the worst-case runtime of the following algorithm?

(a) $O(n)$

(b) $O(n^2)$

(c) $O(\log n)$

(d) $O(\sqrt{n})$

C++

```
// For question 8
void f(int* m, int m_size) {
    for (int i = 0; i * i < n; i++) {
        for (int j = 0; j * j < n; j++) {
            // Some O(1) operation
        }
    }
}
```

9. Pointer Question (MC)

- What is the output of the following code?

(a) 28 12 3

(b) 41 17 7

(c) 48 24 11

(d) 40 12 7

```
// For question 9
int g(int* n, int m) {
    *n += 12;
    m = 6;
    return *n + 4 * m;
}

int f(int* n, int& m) {
    m += 4;
    *n = 5;
    return g(n, m);
}

int main() {
    int n = 12;
    int m = 3;
    std::cout << f(&n, m) << " " << n << m << std::endl;
}
```

Explanation: When `n` is passed into `f`, it is reassigned to 5. Then when it is passed to `g`, we add 12 to it so that it's 17. When `m` is passed to `f`, we add 4 to it so it becomes 7. Since we're not either passing a **pointer** of `m` into `g` nor passing `m` as a **reference** into `g`, **it remains unchanged**. So `m` results in 7. When we're calling `g`, we're passing in the values `g(5, 7)`. We then update `n` to 17 and `m` is changed to 6 in the function. Then it returns `17 + 4 * 6 = 41`. Therefore, our answer is (b)

10. Probability Question (short answer)

- Suppose 100 people all toss a hat into a box and then proceed to randomly pick out a hat. What is the expected number of people who get their own hat back?

Explanation: The trick to this question is to actually consider each person taking a hat from the box of 100 hats as an independent event such that $P(E_1) = P(E_2) = \dots = P(E_{100}) = 1/100$

Furthermore, we can model the random variable $E_i = \begin{cases} 1 & \text{if person } i \text{ finds hat} \\ 0 & \text{otherwise} \end{cases}$

Therefore, our expected number of people who get their own hat back will be

$$1 \cdot P(E_1) + 1 \cdot P(E_2) + \dots + 1 \cdot P(E_{100}) = \sum_{i=1}^{100} P(E_i) = 1$$

- Move Zeroes: Given an array `nums`, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements. Do this in-place. Furthermore, the optimal algorithm should run in

$\Theta(n)$.

Example: [0, 2, 0, 1, 0] -> [2, 1, 0, 0, 0]

```
void moveZeroes(int nums[], int n) {
    int j = 0;
    for (int i = 0; i < n; i++) {
        if (nums[i] != 0) {
            nums[j++] = nums[i];
        }
    }

    for (int i = j; i < n; i++) {
        nums[i] = 0;
    }
}
```

C++

12. Recursion: Given an array nums, find the length of the longest sequence of zeroes recursively. (Hint: You are allowed to use the `std::max` function from STL.)

Example: maxZeroLength([0, 0, 1, 0, 0, 0], 6, 0) = 3

```
int maxZeroLength(int nums[], int len, int startIdx) {
    if (startIdx == len) {
        return 0;
    }
    int maxLen = 0;
    while (startIdx < len && nums[startIdx++] == 0) {
        maxLen++;
    }
    return std::max(maxLen, maxZeroLength(nums, len, startIdx));
}
```

C++