Solution 3

Group A

1) PROBLEM 2. In general, *independence* implies *uncorrelated*, but not vice versa.

   E.g. Two random variables X and Y are independent if and only if $F_{X,Y}(x,y) = F_X(x)F_Y(y)$. They are uncorrelated if and if $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$. Assuming independence, it is easy to check that

   $$\mathbb{E}[XY] = \int xydF_{X,Y}(x,y) = \int xdF_X(x) \cdot \int ydF_Y(y) = \mathbb{E}[X]\mathbb{E}[Y].$$

   Moreover, it can be shown that for any two measurable functions f and g where $\mathbb{E}[|f(X)|] < \infty$ and $\mathbb{E}[|g(Y)|] < \infty$, we have

   $$\mathbb{E}[f(X)g(Y)] = \mathbb{E}[f(X)]\mathbb{E}[g(Y)].$$

   Now consider random variables $X = \mathbf{1}_{[0,1/4)} - \mathbf{1}_{[1/4,1/2)}$ and $Y = \mathbf{1}_{[1/2,3/4)} - \mathbf{1}_{[3/4,1)}$ on the probability space $\Omega = [0,1)$ with standard Lebesgue measure. We have $\mathbb{E}[X] = 0$, $\mathbb{E}[Y] = 0$, and $\mathbb{E}[XY] = \mathbb{E}[0] = 0$, so they are uncorrelated. However, $\mathbb{E}[X^2] = 1/2$, $\mathbb{E}[Y^2] = 1/2$, but $\mathbb{E}[X^2Y^2] = \mathbb{E}[0] = 0$, so they are not independent.

2) PROBLEM 5. Consider the correlation matrix of X, Y, Z, we have

   $$\begin{pmatrix} 1 & r & 0 \\ r & 1 & r \\ 0 & r & 1 \end{pmatrix}$$

   This is a semi-definite matrix. So we need to have non-negative determinants of all its diagonal submatrics, i.e. $1 - r^2 \geq 0$ and $1 - 2r^2 \geq 0$. Hence, we have

   $$-\frac{\sqrt{2}}{2} \leq r \leq \frac{\sqrt{2}}{2}.$$

3) PROBLEM 12. This is a dynamic programming problem. Let

   global[i][j]: Max profit upto day i when we have completed at most j transactions

   local[i][j]: Max profit upto day i when we have completed at most j transactions, *AND* the last transaction is to sell the stock at day j

   The state transition functions are

   global[i][j] = max(local[i][j], global[i-1][j])

   local[i][j] = max(global[i-1][j-1] + max(diff, 0), local[i-1][j] + diff)

```
#include <iostream>
#include <vector>
#include <algorithm>
```

```
using namespace std;

class Solution {
    int helper(vector<int> &prices) {
        int profit = 0;
        for (int i = 0; i < prices.size() - 1; i++) {
            profit = max(profit, profit + prices[i + 1] - prices[i]);
        }
        return profit;
    }
public:
    int maxProfit(int k, vector<int> &prices) {
        int len = prices.size();
        if (len == 0) { return 0; }
        if (k >= len) { return helper(prices); }

        //  rolling array
        vector<int> max_local(k + 1, 0);
        vector<int> max_global(k + 1, 0);
        int diff;
        for (int i = 0; i < len - 1; i++) {
            diff = prices[i + 1] - prices[i];
            for (int j = k; j >= 1; j--) {
                max_local[j] = max(max_global[j - 1] + max(diff, 0), max_local[j] + diff);
                max_global[j] = max(max_local[j], max_global[j]);
            }
        }
        return max_global[k];
    }
};
```

4) PROBLEM 17. (By Fei He) Idea: first, both robots move slowly to the right (right, left, right) until left robot hit 0 (right robot never hit zero), then left robot speed up (goright() only works for left robot) and finally they will meet

```
while(!at_zero()){
    go_right();
    go_left();
    go_right();
}
go_right();
```