```cpp
/* UVa 11450 - Wedding Shopping - Bottom Up */
#include <cstdio>
#include <cstring>
using namespace std;

int main() {
  int i, j, k, TC, M, C;
  int price[25][25];                    // price[g (<= 20)][model (<= 20)]
  bool reachable[25][210];    // reachable table[g (<= 20)][money (<= 200)]
  scanf("%d", &TC);
  while (TC--) {
    scanf("%d %d", &M, &C);
    for (i = 0; i < C; i++) {
      scanf("%d", &price[i][0]);               // we store K in price[i][0]
      for (j = 1; j <= price[i][0]; j++) scanf("%d", &price[i][j]);
    }

    memset(reachable, false, sizeof reachable);         // clear everything
    for (i = 1; i <= price[0][0]; i++)       // initial values (base cases)
      if (M - price[0][i] >= 0)      // to prevent array index out of bound
        reachable[0][M - price[0][i]] = true;  // using first garment g = 0

    for (i = 1; i < C; i++)                   // for each remaining garment
      for (j = 0; j < M; j++) if (reachable[i - 1][j]) // a reachable state
        for (k = 1; k <= price[i][0]; k++) if (j - price[i][k] >= 0)
          reachable[i][j - price[i][k]] = true;   // also a reachable state

    for (j = 0; j <= M && !reachable[C - 1][j]; j++); // the answer in here

    if (j == M + 1) printf("no solution\n");         // last row has on bit
    else            printf("%d\n", M - j);
} } // return 0;
```

```
import java.util.*;

class Main { /* UVa 11450 - Wedding Shopping - Bottom Up */
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int i, j, l, TC, M, C, K;
    int[][] price = new int[25][25]; // price[g (<= 20)][model (<= 20)]
    Boolean[][] reachable = new Boolean[210][25]; // reachable table[money (<= 200)]
[g (<= 20)]

    TC = sc.nextInt();
    while (TC-- > 0) {
      M = sc.nextInt(); C = sc.nextInt();
      for (i = 0; i < C; i++) {
        K = sc.nextInt();
        price[i][0] = K; // to simplify coding, we store K in price[i][0]
        for (j = 1; j <= K; j++)
          price[i][j] = sc.nextInt();
      }

      for (i = 0; i < 210; i++)
        for (j = 0; j < 25; j++)
          reachable[i][j] = false; // clear everything

      for (i = 1; i <= price[0][0]; i++) // initial values
        if (M - price[0][i] >= 0)
          reachable[M - price[0][i]][0] = true; // if only using first garment g = 0

      for (j = 1; j < C; j++) // for each remaining garment (note: this is written in
column major)
        for (i = 0; i < M; i++) if (reachable[i][j - 1]) // if can reach this state
          for (l = 1; l <= price[j][0]; l++) if (i - price[j][l] >= 0) // flag the
rest
            reachable[i - price[j][l]][j] = true; // as long as it is feasible

      for (i = 0; i <= M && !reachable[i][C - 1]; i++); // the answer is in the last
column

      if (i == M + 1)
        System.out.printf("no solution\n"); // nothing in this last column has its
bit turned on
      else
        System.out.printf("%d\n", M - i);
    }
  }
}
```