A: Between Two Integers
３つの整数 A,B,C を入力として受け取ります。問題文の通り、A<=C と C<=B の２つの条件を同時に満たしているか判定を行います。最後に、条件を満たしている場合は「Yes」、そうでない場合は「No」を出力します。

B: Counting Roads
各都市から他の都市に何本の道路が伸びているかを調べます。まず、ループを使い i(1<=i<=N)番目の都市に注目します。次に、ループを用いて全ての道路を調べていき、道路の両端に i 番目の都市が含まれているかを判定し、数えていきます。その後に、数えた道路の本数を i(i<=i<=N)番目の都市から伸びている道路の本数として出力します。これらの操作は、２重ループを用いることで実装ができます。時間計算量は O(NM)となり、これは間に合います。
配列を利用すると、より高速に答えを求めることが可能です。まず、各都市から何本の道路が伸びているかを管理する長さ N の配列 road を用意し、全ての要素を 0 に初期化します。次に、ループを作って全ての道路について調べていきます。この時、ある道路の両端が都市 a と都市 b だったときに、road[a-1]と road[b-1]の値を 1 増やします。最後に、各都市から何本の道路が伸びているかを road を用いて出力します。この解放の時間計算量は、O(N+M)となります。

C: Big Array
まず、この問題で求めたい答えは、入力によって生成される配列の小さい方から K 番目の値です。圧縮された入力を展開して元の配列を求めると、その要素数は最大で 10 の 10 乗となるため MLE となります。
そこで、入力を展開せずに K 番目の値を求める方法を考えます。ここでは、バケツソートを用いた解放について説明します。この解放では、配列の値である ai の範囲は 1 から 10 の 5 乗までと小さいことに注目します。まず、長さ 10 の 5 乗の配列 num を用意し、全ての要素を 0 に初期化します。次に、ループを使って num[ai]に bi を加算します。最後に、ループをもいいて 1 から 10 の 5 乗まで調べていき、K 番目の値を求めます。この解法の時間計算量は O(N+maxA)となるため間に合います。
また配列の代わりに pair を用いたソートでも同じ用に解くことができます。その場合の時間計算量は O(N long N)となるため間に合います。なお、展開後の配列の最大要素数は 10 の 10 乗であるため、32bit 整数型によるオーバーフローに注意してください。

D: Score Attack
まず、問題で与えられるスコアの正負を逆にして、ゲームの最終的なスコアを最小化すると考えてみます。そうすると、この問題は、スコアを距離とみなした頂点 1 から頂点 N への

最短経路問題とみなすことができます。最短経路問題を解くための有名なアルゴリズムには、ダイクストラ法、ベルマンフォード法、ワーシャルフロイド法が存在します。今回の問題では、負のコストの辺が存在するため、ダイクストラ法を適用できません。また、ワーシャルフロイド法の時間計算量は O(N^3)であるため厳しいです(C++なら通る可能性があります)。そこで、時間計算量 O(NM)であるベルマンフォード法をもとに解法を考えていきます。

ここで、最短距離を表す長さ N の配列 dist を用意して、最短距離の 1 回の更新を次のように定義します。

全ての辺に注目して、頂点 a1 の最短距離(dist[ai])とコスト ci から頂点 bi の最短距離(dist[bi])を更新する。

負閉路(辺のコストの総和が負となる閉路)がない場合には、最短距離の更新を N-1 回繰り返すことで最短経路を求めることができます。なぜなら、この最短経路において各頂点は高々 1 回しか登場しないからです(２回以上登場したら閉路ができる)。次に、負閉路の検出について考えてみます。N 回目以降の更新でも最短距離をより短くできれば、その経路上には同じ頂点が 2 回以上登場しているので閉路があると言えます。そして、閉路の存在と最短距離を更新できたことから、負閉路があると言えます。

これらの事実を利用して、次のような解法が考えられます。

1. 頂点 1 の最短距離を dist[1]=0、その他の頂点 v の最短距離を dist[v]=∞と初期化します
2. 最短距離の更新を N-1 回繰り返します(経路の長さは最大で N-1 であるため)
3. 頂点 N の最短距離を表す変数として ans=dist[N]とします
4. 次に、負閉路を検出するための長さ N の配列 negative を用意して、false で初期化します
5. 最短距離の更新を N 回繰り返す(負閉路の長さは最大で N であるため)。ただし、このとき更新された頂点 bi について、negative[bi]=true とします。また、negative[ai]が true の場合には、negative[bi]を true にします

そして、negative[N]が true になっている場合には「inf]、そうでない場合には-ans を出力します。この問題の場合には∞> -N min ci となるように設定すれば十分です。この解法の時間計算量は O(NM)となり、十分間に合います。