Giacomo
Camposampiero

Assessment 1

First submission: 0%

```
int best_win(vect<vect<vect<int>>> &memo, vect<int> &values, int i, int j, int k,
             int m, int move) {
  if(i == j) return 0;
  int move1, move2, state;
  if((move%m) == k ) {
    // player move, maximize the result
    state = memo[i][j][0];
    if(state == -1) {
      // not explored move
      move1 = values[i] + best_win(memo, values, i+1, j, k, m, move+1);
      move2 = values[j] + best_win(memo, values, i, j-1, k, m, move+1);
      state = (move1 > move2)? move1 : move2;
      memo[i][j][0] = state;
    }
  } else {
    // non player move
    state = memo[i][j][1];
    if(state == -1) {
      move1 = best_win(memo, values, i+1, j, k, m, move+1);
      move2 = best_win(memo, values, i, j-1, k, m, move+1);
      state = (move1 > move2)? move2 : move1;
      memo[i][j][1] = state;
    }
  }
  return state;
}

void testcase() {
  int n; std::cin >> n;
  int m; std::cin >> m;
  int k; std::cin >> k;

  std::vector<int> values(n);
  for(int i=0; i<n; i++) std::cin >> values[i];

  std::vector<int> dup(2, -1);
  std::vector<std::vector<int>> col(n, dup);
  std::vector<std::vector<std::vector<int>>> memo(n, col);

  int winnings = best_win(memo, values, 0, n-1, k, m, 0);
  std::cout << winnings << std::endl;
}
```

Second submission: 25%

```
int best_win(vect<vect<vect<int>>> &memo, vect<int> &values, int i, int j, int k,
             int m, int move) {
  // termination cases
  if((i == j) && ((move%m) == k)) return values[i];
  if(i == j) return 0;

  int left, right, state;
  if((move%m) == k ) {
    // player move, maximize the result
    state = memo[i][j][0];
    if(state == -1) {
      left = values[i] + best_win(memo, values, i+1, j, k, m, move+1);
      right = values[j] + best_win(memo, values, i, j-1, k, m, move+1);
      state = (left > right)? left : right;
      memo[i][j][0] = state;
    }
  } else {
    // non player move
    state = memo[i][j][1];
    if(state == -1) {
      left = best_win(memo, values, i+1, j, k, m, move+1);
      right = best_win(memo, values, i, j-1, k, m, move+1);
      state = (left > right)? right : left;
      memo[i][j][1] = state;
    }
  }
  return state;
}

void testcase() {
  int n; std::cin >> n;
  int m; std::cin >> m;
  int k; std::cin >> k;

  std::vector<int> values(n);
  for(int i=0; i<n; i++) std::cin >> values[i];
  std::vector<int> dup(2, -1);
  std::vector<std::vector<int>> col(n, dup);
  std::vector<std::vector<std::vector<int>>> memo(n, col);

  int winnings = best_win(memo, values, 0, n-1, k, m, 0);
  std::cout << winnings << std::endl;
}
```

Third submission: 100%

```
int best_win(vect<vect<int>> &memo, vect<int> &values, int i, int j, int k, int m,  int move)
  // termination cases
  if(i == j) {
    memo[i][j] = ((move%m) == k)? values[i] : 0;
    return memo[i][j];
  }
  if(memo[i][j] != -1) return memo[i][j];
  int res1 = memo[i+1][j], res2 = memo[i][j-1];

  if(res1 == -1) res1 = best_win(memo, values, i+1, j, k, m, move+1);
  if(res2 == -1) res2 = best_win(memo, values, i, j-1, k, m, move+1);

  int state;
  if((move%m) == k) {
    if(res1 + values[i] > res2 + values[j]) state = res1 + values[i];
    else state = res2 + values[j];
  } else state = (res1 > res2)? res2 : res1;

  memo[i][j] = state;
  return state;
}

void testcase() {
  int n; std::cin >> n;
  int m; std::cin >> m;
  int k; std::cin >> k;

  std::vector<int> values(n);
  for(int i=0; i<n; i++) std::cin >> values[i];
  std::vector<std::vector<int>> memo(n, std::vector<int>(n, -1));

  int winnings = best_win(memo, values, 0, n-1, k, m, 0);
  std::cout << winnings << std::endl;
}
```