{

    "": {

        "prefix": "graph\_segment\_tree",

        "body": [

          "",

          "const int segSize=1e5+5;",

          "vector<int> a(segSize), seg(4 \* segSize);",

          "",

          "void build(int index, int low, int high)",

          "{",

          "    if(low == high)",

          "    {",

          "        seg[index] = a[low];",

          "        return;",

          "    }",

          "",

          "    int mid = (low + high) / 2;",

          "    build(2\*index+1, low, mid); //left side of the tree",

          "    build(2\*index+2, mid+1, high); //right side of the tree",

          "",

          "    //the below expression can be changed according to the ques:",

          "    seg[index] = max(seg[2\*index+1], seg[2\*index+2]); //--> here this expression gives the max element from index l to r",

          "}",

          "",

          "",

          "int query(int index, int low, int high, int l, int r)",

          "{",

          "    if(low>=l && high<=r)",

          "    {",

          "        return seg[index];",

          "    }",

          "    if(high<l || low>r)",

          "    {",

          "        return INT\_MIN;",

          "    }",

          "",

          "    int mid = (low + high) / 2;",

          "    int left = query(2\*index+1, low, mid, l, r);",

          "    int right = query(2\*index+2, mid+1, high, l, r);",

          "",

          "    return max(left, right);",

          "}",

          "",

          "",

          "void segment\_tree()",

          "{",

          "    //taking input:",

          "    int n;",

          "    cin>>n;",

          "",

          "    for(int i=0; i<n; i++)",

          "    {",

          "        cin>>a[i];",

          "    }",

          "",

          "    build(0, 0, n-1);",

          "",

          "    //taking query:",

          "    int q;",

          "    cin>>q;",

          "",

          "    while(q--)",

          "    {",

          "        int l, r;",

          "        cin>>l>>r;",

          "",

          "        cout<<query(0, 0, n-1, l, r)<<endl;",

          "    }",

          "}",

          ""

        ],

        "description": ""

      }

}