**Experiment:-7**

**StudentName**:Neetu **UID**22BCS17083

**Branch:** CSE **Section/Group**22BCS\_IOT-641/A

**Semester:**6th **DateofPerformance:**18/03/2025

**SubjectName:**AdvancedProgrammingLab-2**SubjectCode:**22CSP-351

# Problem-1

1. **Aim:**ClimbingStairs

# Objective:

* + **Understanding the Problem:** To understand how to solve the staircase problem using a simple mathematical pattern based on previous steps.
  + **Using Fibonacci Sequence:** To learn how the Fibonacci sequence helps calculate the number of ways to climb stairs.
  + **Practicing Code Efficiency:** To practice using loops and variables to write efficient and clean code by updating values and avoiding repetitive calculations, which improves performance.
  + **HandlingEdgeCases:**Tounderstandhowtohandleedgecaseslikesmallvaluesandensure correct output.
  + **Improving Problem-Solving:** To improve problem-solving skills by applying dynamic programming concepts

# Implementation/Code:

classSolution{ public:

intclimbStairs(intn){ if (n==1) return 1; int a=1,b=2;

for(inti=3;i<=n;i++){ int temp =a+b;

a=b; b=temp;

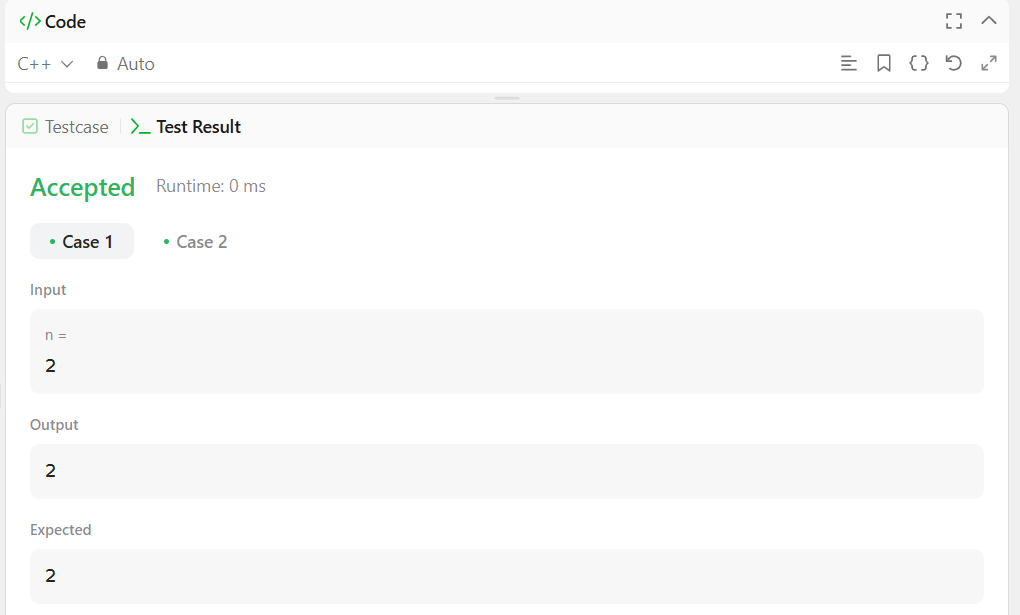
}

return b;

}

};

# Output:

****

**Figure1**

# LearningOutcome:

* + **PatternRecognition:**Youwillunderstandhowtofindpatternsandusethemtosolve coding problems.
  + **ApplyingFibonacci:**YouwilllearnhowtoapplytheFibonaccisequenceinreal-life scenarios.
  + **EnhancingCodingSkills:**Youwillimproveyourcodingskillsbypracticingloopsand updating variables.
  + **SimplifyingProblems:**Youwillunderstandhowtosimplifyproblemsbybreakingthem into smaller steps.
  + **BuildingConfidence:**Youwillbecomemoreconfidentinsolvingmathematicalproblems using code.

# Problem-2

1. **Aim:**MaximumSubarray

# Objectives:

* + **UnderstandingtheProblem:**Tounderstandhowtofindthesubarraywiththelargestsum from a given integer array using a logical approach.
  + **UsingKadane’sAlgorithm:**TolearnhowKadane’salgorithmhelpsfindthemaximum subarray sum by efficiently updating current and maximum sums.
  + **Practicing Efficient Coding:**To practice usingloopsand conditionstoupdate thesum quickly and avoid unnecessary calculations.
  + **Handling Negative Numbers:** To understand how to handle both positive and negativevalues while calculating the maximum sum.
  + **ExploringAdvancedApproaches:**Toexplorethedivideandconquermethodforsolving the problem more efficiently with deeper understanding.

# Implementation/Code:

classSolution{ public:

intmaxSubArray(vector<int>&nums){

intmaxSum=nums[0],currentSum=nums[0]; for (int i = 1; i < nums.size(); ++i) {

currentSum=max(nums[i],currentSum+nums[i]); maxSum = max(maxSum, currentSum);

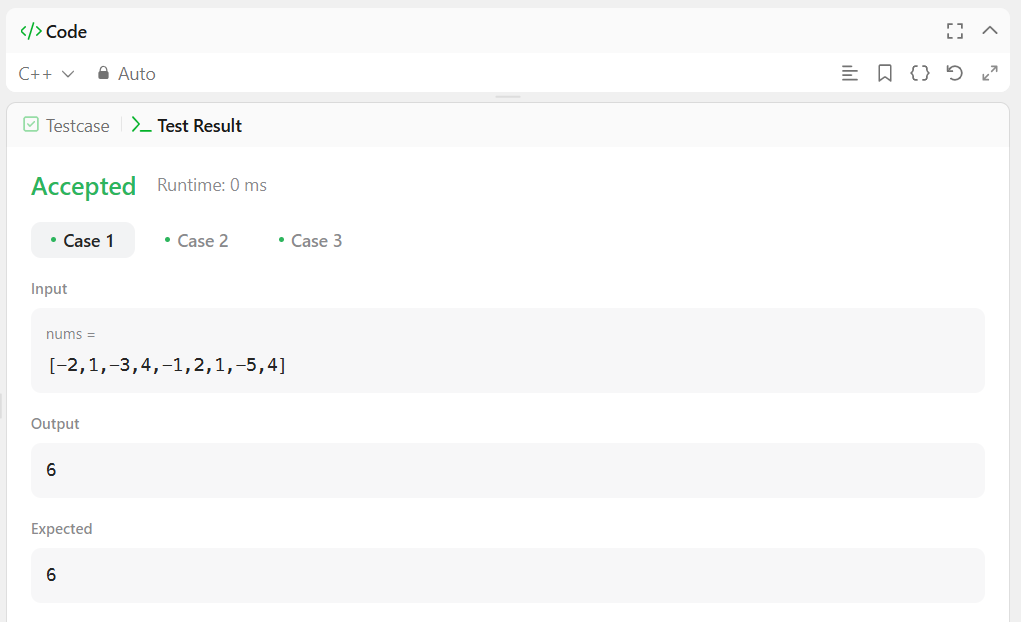
}

returnmaxSum;

}

};

# Output:

****

**Figure2**

# LearningOutcomes

* + **BetterProblem-SolvingSkills:**Youwilllearnhowtoanalyzearrayproblemsanddevelop a logical approach to find the largest sum.
  + **UnderstandingKadane’sAlgorithm:**YouwillunderstandhowKadane’salgorithmworks and why it is effective for finding maximum subarray sums.
  + **WritingCleanandFastCode:**Youwillimproveyourabilitytowriteefficientcodeby properly using loops and conditions.
  + **HandlingEdgeCases:**Youwillbeabletohandlecaseswithmixedpositiveandnegative numbers confidently.
  + **ApplyingAdvancedMethods:**Youwillgainexperienceinusingthedivideandconquer approach to solve complex array problems.

# Problem:-3

1. **Aim:**Jump Game

# Objectives:

* + **UnderstandingtheProblem:**Tounderstandhowtocheckifyoucanreachthelastindex using jump values in the array.
  + **UsingGreedyApproach:**Tolearnhowthegreedyapproachhelpsinfindingthemaximum reachable index at each step.
  + **PracticingEfficientCode:**Topracticewritingefficientcodeusingloopsandconditionsto reduce calculations.
  + **HandlingStuckPositions:**Tounderstandhowtohandlecaseswhereprogressisblocked due to zero jump value.
  + **OptimizingPerformance:**Toimproveperformancebystoppingearlyoncethelastindex is reachable.

# Implementation/Code:

classSolution{ public:

boolcanJump(vector<int>&nums){ int maxReach=0;

for(inti=0;i<nums.size();i++){ if(i>maxReach)returnfalse;

maxReach=max(maxReach,i+nums[i]);

if(maxReach>=nums.size()-1)return true;

}

returnfalse;

}

};

# Output:

**Figure3**

# LearningOutcomes:

* + **BetterProblemSolving:**Youwilllearnhowtosolvearray-basedmovementproblemsstep- by-step.
  + **UnderstandingGreedyMethod:**Youwillunderstandhowthegreedyapproachhelpsin making the best jump decision.
  + **WritingCleanCode:**Youwillimproveyourcodingskillsbywritingsimpleandoptimized code.
  + **HandlingEdgeCases:**Youwillknowhowtohandlecaseswheremovementisblockedby zero jump value.
  + **ImprovingEfficiency:**Youwilllearnto writefastersolutionsbyreducingunnecessary calculations.