Experiment:-8

StudentName: Sushil Kumar **UID:**22BCS16123

Branch: CSE **Section/Group:**22BCSIOT-603/B

Semester:6th **DateofPerformance:**3/4/2025

SubjectName: AdvancedProgrammingLab-2 **Subject Code:**22CSP-351

Problem-1

1. **Aim:**MaxUnits onaTruck

2. **Objective:**

- **Optimizeloadingofboxesontoatruck:**Learnhowtomaximizethetotalunitsofboxesthat can be loaded given a truck's size limit, applying strategies to make the best use of available space.
- **Sortboxesbyunitsperbox:**Understandhowsortingboxesbasedonthenumberofunitsper box can help prioritize which boxes to load first, ensuring the most valuable boxes are placed on the truck.
- **Applygreedy algorithm techniques:** Gain hands-on experience with greedy algorithms, whichmakelocallyoptimalchoicesateachstep,toachievetheglobalmaximumofunitsloaded on the truck.
- **Work with 2D arrays and loops:** Improve your ability to handle and manipulate 2D arrays, as well as use loops and conditionals to process data efficiently in coding tasks.
- **Handlespaceconstraintsandoptimization:**Learnhowtomanagesituationswherespaceis limited and how to optimize the use of resources, like loading boxes in themost efficient way possible.

3. **Implementation/Code:**

```
classSolution{
public:
   int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
      sort(boxTypes.begin(),boxTypes.end(),[](vector<int>&a,vector<int>&b){
         returna[1]>b[1];
      });
      inttotalUnits =0, i=0;
      while(truckSize>0&&i<boxTypes.size()){ if
         (boxTypes[i][0] <= truckSize) {
            totalUnits+=boxTypes[i][0]*boxTypes[i][1]; truckSize
            -= boxTypes[i][0];
         }else {
            totalUnits+=truckSize*boxTypes[i][1];
            truckSize = 0;
```

```
            }
            i++;
        }
        return totalUnits;
    }
};
```

4. Output

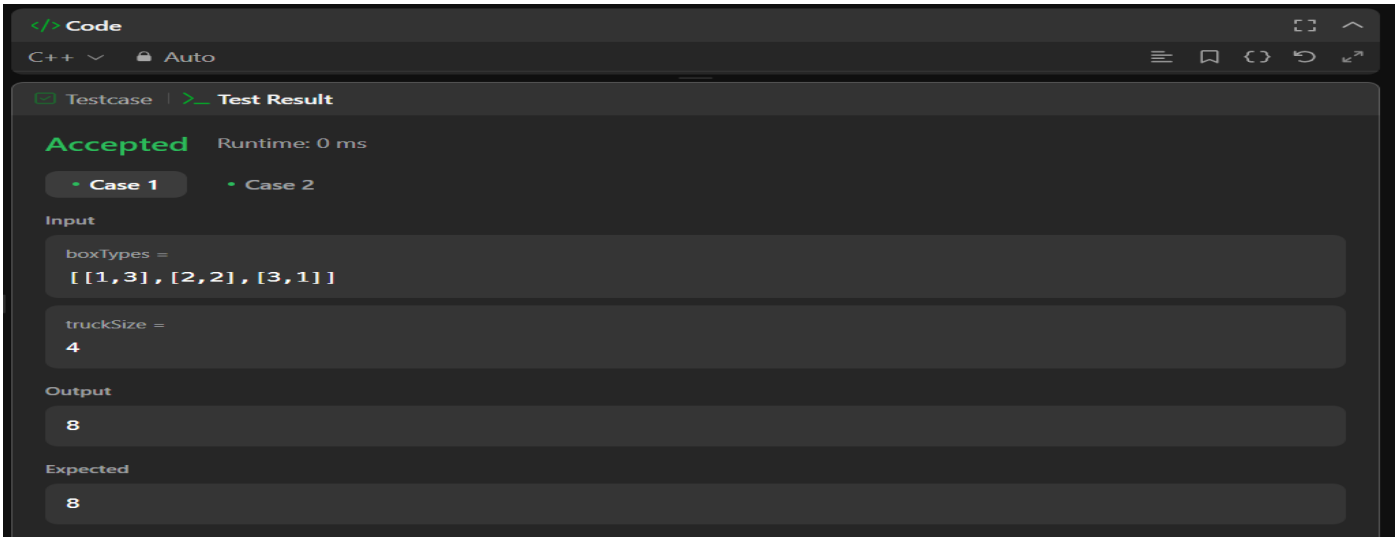

Figure1

5. LearningOutcomes:

- **Efficient sorting and data processing:** Develop a clear understanding of sorting data based on specific criteria (like units per box) to solve real-world optimization problems effectively.
- **Calculating totals with loops and conditions:** Master the use of loops and conditional statements to calculate totals, ensuring correct results even with varying input sizes and constraints.
- **Handlingedgecases:**Learnhowtodealwithdifferentedgecases,suchaswhenthetruckruns out of space or there are more boxes than available space.
- **Strengthen problem-solving skills:** Enhance your ability to break down complex problems into simpler steps, applying algorithms and logic to find efficient solutions.
- **Optimize resource allocation:** Gain experience in maximizing resource use, such as truck space, by applying strategies that ensure the best possible use of available resources.

Problem-2

1. **Aim:**MinOperationstomakearrayincreasing.
2. **Objectives:**

- **Makeanarraystrictlyincreasing:**Learnhowtomodifyanarraysothateachnumberis greater than the previous one by making the fewest changes.
- **Usethe smallest number of operations:** Understand how to increment elements efficiently to achieve the required increasing order with minimal changes.
- **Apply logic to find differences:** Learn how to compare consecutive elements and calculatehow much an element needs to increase to maintain strict order.
- **Workwithloopsandconditionals:**Improveprogrammingskillsbyusingloopsand conditions to check and update elements in an array.
- **Solve real-world optimization problems:** Understand how to optimize solutions by making the smallest possible changes to meet given constraints.

3. Implementation/Code:

```
classSolution{
public:
   intminOperations(vector<int>&nums){ int
      operations = 0;
      for(inti=1;i<nums.size();i++){ if
         (nums[i] <= nums[i - 1]) {
            intdiff=nums[i-1]-nums[i]+1;
            nums[i] += diff;
            operations+=diff;
         }
      }
      return operations;
   }
};
```

4. Output:

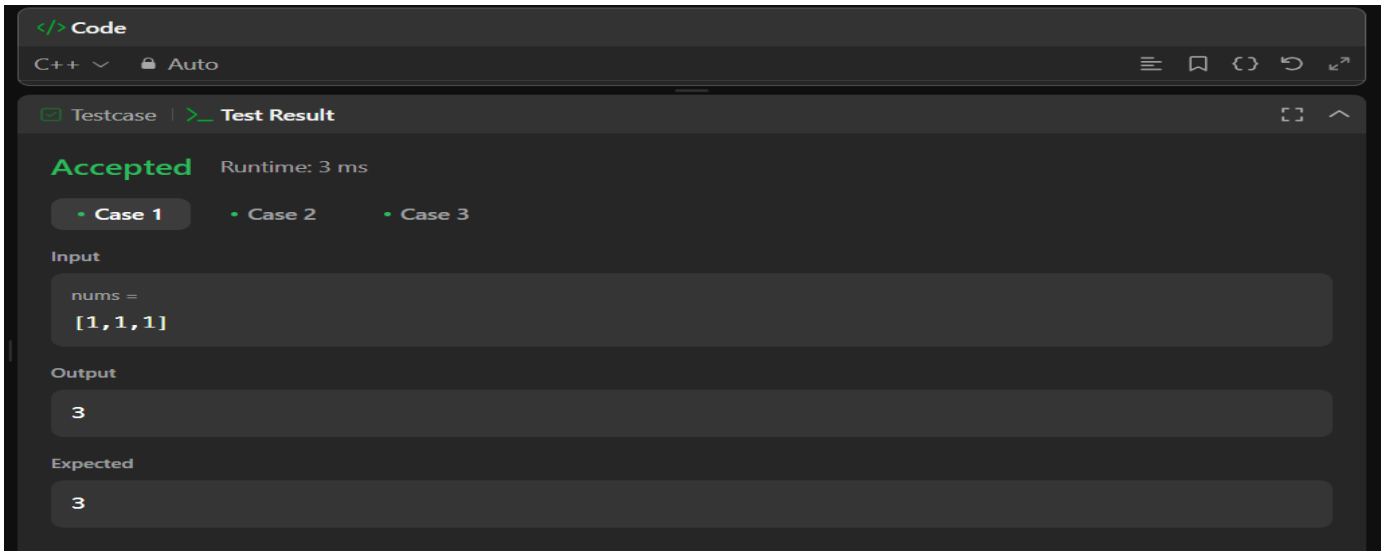

Figure2

5. LearningOutcomes:

- **Understandarraymodifications:**Gaintheabilitytoanalyseandupdateanarraytomeet specific conditions using the least number of operations.
- **Useloopstocheckandadjustvalues:**Developskillsinusingloopsandconditionalsto compare and modify elements efficiently.
- **Optimizeproblem-solvingstrategies:**Learnhowtofindthesmallestnumberofchanges needed to achieve a required goal in an algorithm.
- **Handleedge cases in constraints:** Be able to manage situations where numbers are already increasing or require multiple adjustments.
- **Improvealgorithmicthinking:**Strengthenproblem-solvingskillsbyapplyinglogical reasoning and efficient strategies to achieve the best result.

Problem:-3

1. Aim:MaxScorefromremovingsubstrings

2. Objectives:

- Remove specific substrings for maximum points: Learn how to remove "ab" and "ba" from a string to earn the highest possible score by applying the best order of operations.
- Use stack-based string processing: Understand how to efficiently remove substrings using astack approach, making the process faster and more structured.
- Comparedifferentoperationorders:Learnhowchoosingtherightsequenceofremovals(based on points assigned) can maximize the final score.
- Optimizestringmanipulation:Improveproblem-solvingskillsbyhandlinglargestringsefficiently without unnecessary operations or extra memory usage.
- Applygreedyalgorithmconcepts:Understandhowagreedyapproachhelpsinmakingthebest choice at each step to achieve the maximum total score.

3. Implementation/Code:

```
classSolution{
public:
   intmaximumGain(strings,intx,inty){ int
      score = 0;
      if(x >y) {
         score+=removePair(s,'a', 'b', x);
         score+=removePair(s,'b', 'a', y);
      }else {
         score+=removePair(s,'b', 'a', y);
         score+=removePair(s,'a', 'b', x);
      }
      returnscore;
   }
   intremovePair(string&s,charfirst,charsecond,intpoints){ string
      temp = "";
```

```
            int score = 0;
            for(charc:s){
               if(!temp.empty()&&temp.back()==first&&c==second){ temp.pop_back();
                  score+=points;
               }else {
                  temp.push_back(c);
               }
            }
            s = temp;
            returnscore;
         }
};
```

4. Output:

Figure3

5. LearningOutcomes:

- **Understand substring removal strategies:** Gain the ability to remove specific pairs from a string while maintaining efficiency and correctness.
- **Improveproblem-solvingwithstacks:**Learnhowtouseastack-likemethodtokeeptrackof character sequences and remove pairs dynamically.
- **Develop logical thinking for optimization:** Understand how to determine the best order ofoperations to achieve the highest possible score.
- **Handle large input sizes efficiently:** Learn how to manage operations on long strings while keeping execution time within acceptable limits.
- **Strengthenalgorithmicskills:**Improvetheabilitytodesignandimplementefficient algorithms that maximize output while minimizing computational cost.