| **STUDENT PORTFOLIO** | |
| --- | --- |
| Insert Photo | **Name: P.Masood**  **Register Number:RA2311030010313**  **Mail ID:pm6387@srmist.edu.in**  **Department:Networking and communications**  **Semester: IV**  **Email: pmasood935@gmail.com**  **Linkedin:** [**www.linkedin.com/in/p-masood-363822319**](http://www.linkedin.com/in/p-masood-363822319)  **Github: https://github.com/Masood090** |
| **Subject Title: 21CSC204J Design and Analysis of Algorithm**  **Handled By: Dr.M.Jeyaselvi**  E-Lab Completion Status  Circle of Execution | |
| Explanation of at least one program  Uploaded image  This C++ program solves the sandwich grid problem described in the image. Let's break it down step-by-step.  🔍 Problem Summary  You are given a grid of characters (some cells with letters and others with ?). Each letter is a unique student’s initial, and each student wants a single rectangular region (grid-aligned) with only their letter. Surya has to fill in the ? cells to extend each letter into a rectangle — one for each student.  It is guaranteed that a solution exists.  🧠 How the Code Works  1. Input Reading  cpp  CopyEdit  int R, C;  cin >> R >> C;  vector<string> g(R);  for (int i = 0; i < R; i++) cin >> g[i];   * Reads the number of rows (R) and columns (C) * Reads the grid into a vector of strings g.   2. Horizontal Fill (Left to Right, then Right to Left)  cpp  CopyEdit  for (int i = 0; i < R; i++) {  char last = '?'; int j = 0;  while (j < C) {  if (g[i][j] == '?') g[i][j] = last;  else last = g[i][j];  j++;  }  last = '?'; j = C - 1;  while (j >= 0) {  if (g[i][j] == '?') g[i][j] = last;  else last = g[i][j];  j--;  }  }  This fills in the ? from:   * Left to right using the most recent seen letter (last) * Then right to left to catch any leading/trailing ?s in the row   3. Vertical Fill (Top to Bottom, then Bottom to Top)  cpp  CopyEdit  for (int j = 0; j < C; j++) {  char last = '?'; int i = 0;  while (i < R) {  if (g[i][j] == '?') g[i][j] = last;  else last = g[i][j];  i++;  }  last = '?'; i = R - 1;  while (i >= 0) {  if (g[i][j] == '?') g[i][j] = last;  else last = g[i][j];  i--;  }  }  This ensures that if some rows were initially empty, the vertical pass fills them using letters from above/below.  4. Output the Grid  cpp  CopyEdit  cout << "Line #" << T << ":\n";  for (auto &row : g) cout << row << "\n";  Prints the modified grid with all ? replaced.  5. Driver Function  cpp  CopyEdit  int main() {  int T;  cin >> T;  for (int t = 1; t <= T; t++) solve(t);  }  Processes multiple test cases.  ✅ Guarantees the Rectangular Constraint  The horizontal then vertical fill ensures that each letter forms a solid block, and no letter mixes with another — hence it respects the rectangular region constraint. | |
| **Lab Experiment Completion status** | |
| **REAL WORLD APPLICATION IN DAA PPT VR/SIMULATION DEMO** | |
| **NPTEL/HOTS Questions Solution.**  **Program Explanation**  Problem Description: Laasya bought a new volleyball in the sports shop. It looks like a medium size. She somehow found the radius of the sphere. But she would like to know the volume of that ball. Can you help her in finding the volume of the ball?  Functional Description: Volume = (4.0/3.0) × π × r³, π = 3.14  Constraint: 1.00 ≤ r ≤ 5.00  Input Format: The only line of input has a single value of type float representing the radius of the ball.  Output Format: Print the volume of the ball in a single line.  Code:  #include int main() { float radiusofball, volumeofball; scanf("%f", &radiusofball); volumeofball = (4.0/3.0) \* radiusofball \* radiusofball \* radiusofball \* 3.14; printf("%f", volumeofball); return 0; }  Explanation:  Line-by-Line Explanation:   * 1. #include : Includes the Standard Input Output library for basic I/O operations. * 2. int main(): The main function where program execution begins. * 3. Variable Declaration: float radius, volume;   + radius: Stores the input value (ball's radius).   + volume: Will store the calculated volume. * 4. Input Reading: scanf("%f", &radius);   + %f: Format specifier for float input.   + &radius: Stores the input value in the radius variable. * 5. Volume Calculation: volume = (4.0f / 3.0f) \* 3.14f \* radius \* radius \* radius;   + Uses the sphere volume formula: V = (4/3)πr³.   + 4.0f/3.0f: The fractional part of the formula.   + 3.14f: Approximate value of π (pi).   + Multiplied by radius cubed (radius × radius × radius). * 6. Output: printf("%.2f", volume);   + %.2f: Displays the float with exactly 2 decimal places.   + Prints the calculated volume. * 7. return 0;: Indicates successful program termination.   Key Features:   * Precision Handling: Uses float variables with f suffix (e.g., 3.14f) for consistent floating-point arithmetic. * Formula Implementation: Correctly implements the mathematical formula for sphere volume. * Order of operations is maintained with parentheses. * Input/Output: Simple console interface, clean output formatting.   Example Execution:  Input: 3.0  Calculation: (4.0/3.0) × 3.14 × 3.0 × 3.0 × 3.0 = 1.333... × 3.14 × 27 = 113.04  Output: 113.04 | |
| **Certifications Based on DAA**      **Codeforces Completion**        **Signature** | |