Functions in the Program:

1. **main()**:
    - Starts the game by calling rngame() and endScreen().
2. **rngame()**:
    - Main game loop, handles events, updates the state of the game, and manages the flow of the game, including bubbles and arrows, along with music.
3. **mkeblkbrd()**:
    - Initializes the game board (array) filled with empty values (blank).
4. **setbb(array, gameclrlist)**:
    - Sets up the bubbles on the board with random colors.
5. **setarrpos(array)**:
    - Calculates the positions of each bubble in the grid based on its row and column.
6. **delextrbb(array)**:
    - Deletes any extra bubbles that are placed outside the boundaries of the game screen.
7. **updtclrlist(bbarr)**:
    - Updates the list of bubble colors based on the remaining bubbles on the board.
8. **chkfflotrs(bbarr)**:
    - Checks for floating bubbles in the array and removes them if necessary.
9. **popflotrs(bbarr, cpyofbrd, col, row=0)**:
    - Recursively checks and removes floating bubbles from the game board.

10. **stbb(bbarr, newbb, launchbb, score)**:
    - Handles the logic for shooting a bubble, checking for collision with existing bubbles, and updating the game state.
11. **addbbtotop(bbarr, bubble)**:
    - Adds the newly launched bubble to the top of the board.
12. **popbb(bbarr, row, col, color, dellst)**:
    - Checks and pops connected bubbles of the same color, recursively updating the board and the list of bubbles to be deleted (dellst).
13. **drawbbary(array)**:
    - Draws all the bubbles on the board.
14. **makeDisplay()**:
    - Initializes the game display (surface) and returns the surface and rectangle object.
15. **terminate()**:
    - Exits the game by quitting Pygame and the system.
16. **covnxtbb()**:
    - Clears the space where the next bubble is shown to make way for the new one.
17. **endScreen(score, winorlose)**:
    - Displays the end screen with the player's score and a message indicating whether they won or lost.

Data Structures Used:

1. **Lists**:
    - **bbarr**: A 2D list (grid) representing the game board, where each cell can contain a bubble object or blank.

- **clrlist**: A list of colors representing the different bubble colors.
- **gameclrlist**: A shuffled copy of clrlist used to randomly assign colors to bubbles.
- **dellst**: A list used to store the coordinates of bubbles that need to be popped (deleted).
- **musclist**: A list of music tracks played during the game.
- **fbblist**: A list of floating bubbles that have not been popped yet.

2. **Tuples**:
   - **(row, col)**: Used in several functions to represent the coordinates of a bubble on the board (e.g., dellst stores the (row, col) positions of bubbles to be deleted).
   - **(strx, strY)**: The starting position for the arrow and bubbles on the screen.
3. **Objects (Pygame sprite objects)**:
   - **Bubble**: A class that defines a bubble object. It holds properties like color, radius, position, and angle and includes methods for drawing and updating the bubble's position.
   - **Ary**: A class representing the arrow. It handles the angle and movement of the arrow used to shoot the bubbles.
   - **Score**: A class used to track the score and display it on the screen.
4. **pygame.Rect**:
   - Used for positioning and defining the boundaries of the bubbles and the arrow. pygame.Rect objects are used to store the position of objects (e.g., the rect attribute of Bubble and Ary classes).

Algorithms Used:

1. **Collision Detection**:
    - The stbb() function uses pygame.sprite.collide_rect() to check if the newly launched bubble collides with any existing bubbles on the game board. If there is a collision, the bubble is placed on the grid.
2. **Recursive Search (Flood Fill)**:
    - The popbb() and popflotrs() functions use a **recursive algorithm** to find and remove bubbles that are connected to the current bubble of the same color. This is a common flood fill algorithm similar to the one used in image processing.
3. **Bubble Launch and Trajectory Calculation**:
    - The movement of the bubbles is calculated in the Bubble.update() method, which uses trigonometry (based on the angle of the arrow) to calculate the x and y velocities. The angle is adjusted to simulate a shooting mechanism, and the bubble's position is updated accordingly.
4. **Shuffling (Randomization)**:
    - The random.shuffle() function is used to randomize the order of bubble colors and music tracks.
5. **Bubble Deletion (Matching & Popping)**:
    - The algorithm checks if there are three or more connected bubbles of the same color (in the stbb() function) and pops them, updating the game state and the score.
6. **Game Flow**:
    - The game flow is controlled by the rngame() function, which continuously updates the game state by processing

events (keyboard inputs) and redrawing the screen until a win or lose condition is met.

Object-Oriented Programming (OOP) Concepts Used:

1. **Classes**:
    - **Bubble**: Represents individual bubbles, with attributes like color, position, and speed, and methods for updating its position and drawing itself on the screen.
    - **Ary**: Represents the arrow that shoots the bubbles. It manages the rotation of the arrow and updates its angle based on user input.
    - **Score**: Represents the player's score and manages the display of the score on the screen.
2. **Encapsulation**:
    - Each class encapsulates its data (e.g., Bubble has properties like color, radius, position, etc.) and behavior (e.g., update(), draw() methods), ensuring that each object is responsible for its own state and actions.
3. **Inheritance**:
    - **pygame.sprite.Sprite** is inherited by both Bubble and Ary classes, allowing them to use Pygame's built-in sprite functionalities like position management, collision detection, and drawing.
4. **Polymorphism**:
    - The update() method in both Bubble and Ary classes has similar names but different implementations depending on the object. This demonstrates polymorphism, where the same method name behaves differently for different classes.