
# TRAVEL DESTINATION RECOMMENDATION SYSTEM USING BINARY DECISION TREE

Presented by:
Alicia Theresa Pereira (2462028)
Joann Binny (2462088)
Stacy Anna Dsouza (2462156)
3 BTCS AIML C


# PRESENTATION OUTLINE

- OBJECTIVE
- RELEVANCE
- TREE STRUCTURE VISUALIZATION
- CONCEPT APPLIED
- ALGORITHM
- PSEUDOCODE
- INPUT/OUTPUT SCREENSHOTS
- CODE SNIPPET
- RESULT
- CONCLUSION

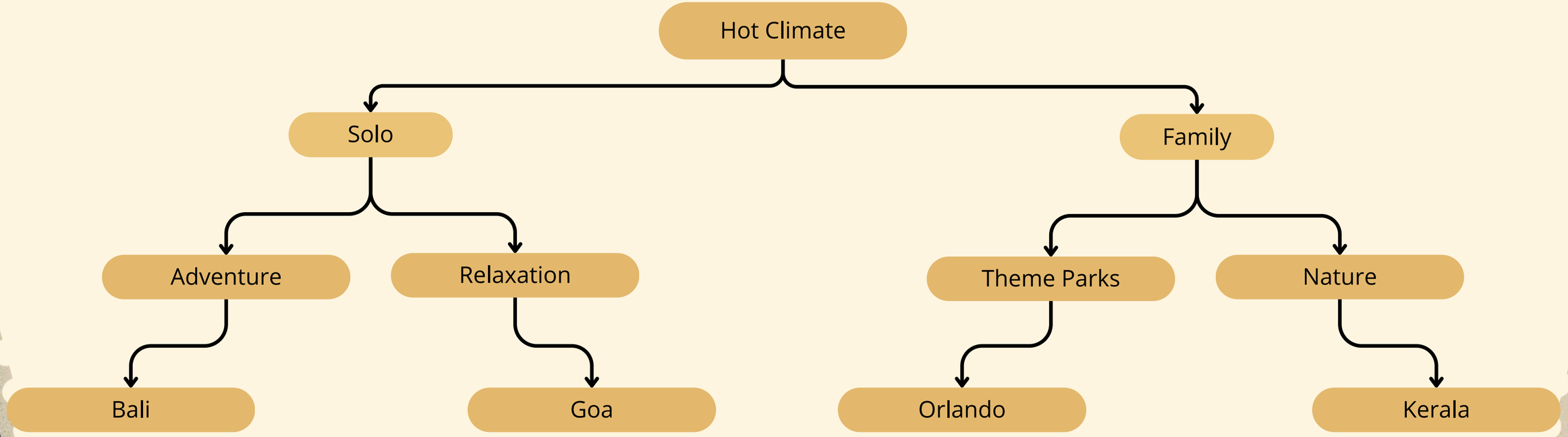
# OBJECTIVE

- Build an interactive travel recommender in C using a binary decision tree.
- Guide users with questions to produce personalized destination suggestions.
- Implement dynamic memory allocation for tree nodes.
- Use recursion for tree traversal and decision logic.
- Create a user-friendly command-line interface.
- Reinforce modular design and debugging skills in C.

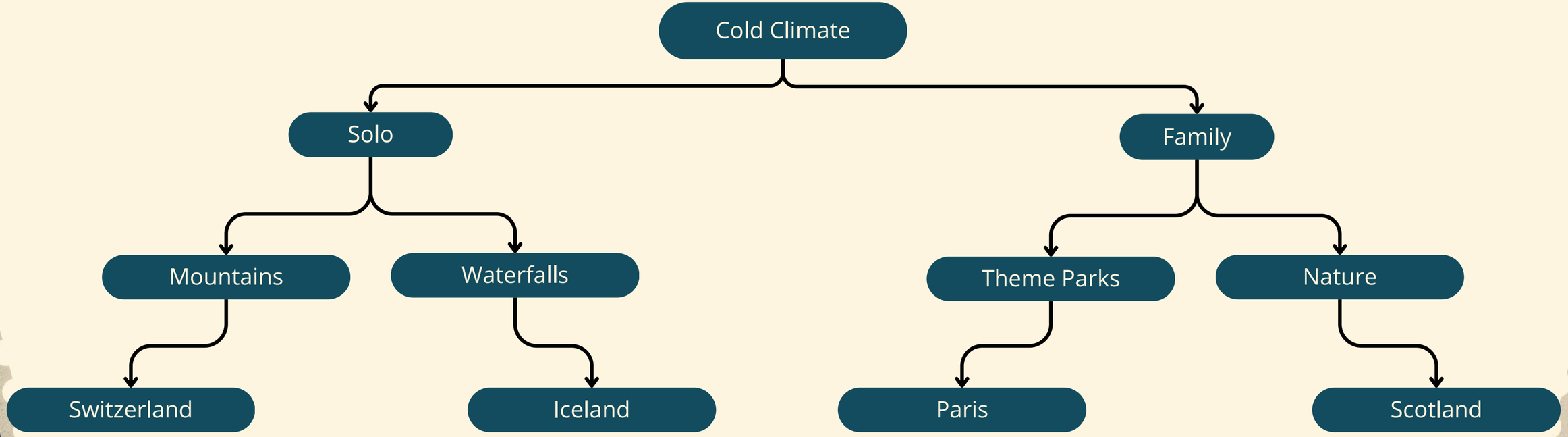

# RELEVANCE

- Applies tree data structures to a real-world problem.
- Teaches memory management (malloc/free) and pointers.
- Strengthens recursion and algorithmic thinking.
- Demonstrates decision-tree logic used in recommender systems.
- Good practice for CLI design and user interaction.
- Useful for portfolio: shows systems design + C programming.
- Encourages teamwork, documentation, and presentation skills.

# TREE STRUCTURE VISUALIZATION



# TREE STRUCTURE VISUALIZATION



# CONCEPT APPLIED

**What is a Binary Tree?**

- Hierarchical structure with at most two children per node (left & right).
- Used here to represent decision-making questions leading to destinations.

**Why Binary Tree for This Project?**

- Binary Decision-Making → Each question has two choices (e.g., Hot vs Cool climate).
- Hierarchical Flow →
  - Level 1: Climate
  - Level 2: Trip Type
  - Level 3: Activity Preference
  - Level 4: Final Destination (Leaf Node)
- Efficient Traversal → Moves step-by-step based on user input until a destination is reached.
- Easy to Implement in C → Structs + pointers make binary trees straightforward.
- Practical Relevance → Mimics how real-world recommender systems work.


# ALGORITHM

1. Start Program
2. Display Welcome Message
3. Ask Climate Choice → Hot (1) / Cool (2)
4. Navigate to Climate Branch
5. Ask Group Type → Solo/Family (1) / Friends/Work (2)
6. Subdivide Further → Solo vs Family, Friends vs Work
7. Ask Preference → Adventure / Relaxation / Other options
8. Traverse Tree Based on User Choices
9. Reach Leaf Node → Destination Recommendation
10. Print Suggested Travel Destination
11. Display Thank-You Message
12. End Program

# PSEUDOCODE

```
START

DISPLAY "Welcome to Travel Recommendation System"
FUNCTION ask(node):
    IF node is NULL:
        RETURN

    IF node.left == NULL AND node.right == NULL:
        PRINT "Recommended Destination: ", node.text
        RETURN
    DISPLAY node.text
    INPUT choice
    IF choice == 1:
        CALL ask(node.left)
    ELSE IF choice == 2:
        CALL ask(node.right)
    ELSE
        PRINT "Invalid choice! Try again."
        CALL ask(node)

MAIN:
    CREATE root node: "Choose Climate: 1. Hot  2. Cool"
    // Build tree structure with questions and destinations
    CONNECT nodes for climate → group type → preferences → destinations
    CALL ask(root)
DISPLAY "Thank you for using the Travel Recommendation System!"

END
```

# INPUT/OUTPUT

```
Travel Recommendation System
==================================

----------------------------------
Choose Climate:
1. Hot/Tropical
2. Cool/Temperate
Enter choice (1 or 2): 1

----------------------------------
Who are you traveling with?
1. Solo/Family
2. Friends/Work
Enter choice (1 or 2): 1

----------------------------------
Choose:
1. Solo
2. Family
```

```
----------------------------
 prefer:
ne Parks/Entertainment
ure/Scenic?
choice (1 or 2): 2

----------------------------
 prefer:
ch Destinations
ntain/Forest?
choice (1 or 2): 1

==============================
ended: Kerala, Hawaii (Beaches)
==============================
```

# INPUT/OUTPUT

```
Travel Recommendation System
====================================

------------------------------------
Choose Climate:
1. Hot/Tropical
2. Cool/Temperate
Enter choice (1 or 2): 1

------------------------------------
Who are you traveling with?
1. Solo/Family
2. Friends/Work
Enter choice (1 or 2): 1

------------------------------------
Choose:
1. Solo
```

```
------------------------------------
Do you prefer:
1. Adventure/Activities
2. Relaxation?
Enter choice (1 or 2): 2

------------------------------------
Do you want:
1. Beach
2. Island Resort?
Enter choice (1 or 2): 2

====================================
Recommended: Santorini, Fiji (Island Resorts)
====================================
```

# CODE SNIPPET

```
typedef struct node {
    char text[500];
    struct node *left;
    struct node *right;
} node;

node* makeNode(const char* str) {
    node* temp = (node*)malloc(sizeof(node));
    if (!temp) {
        printf("Memory error!\n");
        exit(1);
    }
    snprintf(temp->text, sizeof(temp->text), "%s", str);
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

void ask(node* root) {
    if (root == NULL) return;

    if (root->left == NULL && root->right == NULL) {
        printf("\n===================================\n");
        printf("%s\n", root->text);
        printf("===================================\n");
        return;
    }

    int choice;
    printf("\n-----------------------------------\n");
    printf("%s", root->text);
    printf("\nEnter choice (1 or 2): ");
    scanf("%d", &choice);

    if (choice == 1) ask(root->left);
    else if (choice == 2) ask(root->right);
    else printf("Invalid choice!\n");
}
```

# RESULT

- Travel Recommendation System implemented in C using a binary tree.
- Program displays welcome message and asks user questions.
- User navigates choices → Climate → Group Type → Preferences.
- Binary tree traversal leads to final destination recommendation.
- System works successfully and mimics a real-world recommender.

# CONCLUSION

- Demonstrates practical use of binary trees for decision-making.
- Provides an interactive, user-friendly travel recommendation system.
- Applies key C programming concepts: structs, dynamic memory, recursion.
- Models a simplified recommender system used in real platforms.
- Can be extended with file I/O, GUI, database, or multi-way trees.
- Successfully shows how logical trees enable personalized suggestions.

# THANK YOU!