

Project Title: Splay Tree Password Manager

Developed by

Roll No.:22881A05K7

Name A. Raj Kumar

Class: II B.Tech. CSE-D

Regulation: R22

Under the Guidance of

Dr. Vasantha SV, Associate Professor, CSE Dept.

Github Link: <https://github.com/achiever2004/ads-project>

Problem Statement:

Design and implement a password manager that utilizes Splay Trees to efficiently store and retrieve passwords for various accounts. The password manager should allow users to add, update, delete, and retrieve passwords securely.

Algorithm/ Explanation of Logic:

- **Splay Tree Implementation:**

Implement a Splay Tree data structure to store account information.

Each node in the Splay Tree represents an account with fields for the account name, username, and password.

- **Password Operations:**

Implement functions for adding a new account, updating an existing account, deleting an account, and retrieving a password.

During each operation, splay the corresponding node to the root of the tree to optimize future access to recently accessed accounts.

- **Security Measures:**

Incorporate encryption and secure hashing techniques to store passwords securely.

Implement a master password system to ensure access control.

- **User Interface:**

Develop a simple command-line or graphical user interface for users to interact with the password manager.

Include options for adding, updating, deleting, and retrieving passwords.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Account {
    char account_name[50];
    char username[50];
    char password[50];
    struct Account* left;
    struct Account* right;
} Account;

// Splay Tree Operations
Account* rightRotate(Account* x);
Account* leftRotate(Account* x);
Account* splay(Account* root, char key[]);
Account* insert(Account* root, char account_name[], char username[], char password[]);
Account* search(Account* root, char account_name[]);
Account* delete(Account* root, char account_name[]);
void printInOrder(Account* root);

// Password Manager Operations
void addPassword(Account** root, char account_name[], char username[], char password[]) {
    *root = insert(*root, account_name, username, password);
}
```

```
void updatePassword(Account** root, char account_name[], char username[], char password[]) {
```

```
    *root = delete(*root, account_name);
```

```
    *root = insert(*root, account_name, username, password);
```

```
}
```

```
void deletePassword(Account** root, char account_name[]) {
```

```
    *root = delete(*root, account_name);
```

```
}
```

```
char* retrievePassword(Account* root, char account_name[]) {
```

```
    Account* result = search(root, account_name);
```

```
    if (result != NULL) {
```

```
        return result->password;
```

```
    } else {
```

```
        return NULL;
```

```
    }
```

```
}
```

```
int main() {
```

```
    Account* root = NULL;
```

```
    // Test cases
```

```
    addPassword(&root, "Email", "user@example.com", "securePassword1");
```

```
    addPassword(&root, "Bank", "john_doe", "strongPassword123");
```

```
    addPassword(&root, "Social Media", "user123", "password567");
```

```
    // Print the Splay Tree in order
```

```
    printf("Splay Tree in order:\n");
```

```

printInOrder(root);

// Retrieve a password
char* retrievedPassword = retrievePassword(root, "Bank");
if (retrievedPassword != NULL) {
    printf("\nRetrieved Password for Bank: %s\n", retrievedPassword);
} else {
    printf("\nAccount not found\n");
}

// Update a password
updatePassword(&root, "Bank", "john_doe", "newStrongPassword456");

// Print the updated Splay Tree
printf("\nSplay Tree after updating Bank account:\n");
printInOrder(root);

// Delete an account
deletePassword(&root, "Email");

// Print the Splay Tree after deletion
printf("\nSplay Tree after deleting Email account:\n");
printInOrder(root);

return 0;
}

// Splay Tree Implementation
Account* rightRotate(Account* x) {

```

```

Account* y = x->left;
x->left = y->right;
y->right = x;
return y;
}

```

```

Account* leftRotate(Account* x) {
    Account* y = x->right;
    x->right = y->left;
    y->left = x;
    return y;
}

```

```

Account* splay(Account* root, char key[]) {
    if (root == NULL || strcmp(root->account_name, key) == 0) {
        return root;
    }
}

```

```

    if (strcmp(root->account_name, key) > 0) {
        if (root->left == NULL) {
            return root;
        }
    }
}

```

```

    if (strcmp(root->left->account_name, key) > 0) {
        root->left->left = splay(root->left->left, key);
        root = rightRotate(root);
    } else if (strcmp(root->left->account_name, key) < 0) {
        root->left->right = splay(root->left->right, key);
    }
}

```

```

    if (root->left->right != NULL) {
        root->left = leftRotate(root->left);
    }
}

```

```

return (root->left == NULL) ? root : rightRotate(root);
} else {
    if (root->right == NULL) {
        return root;
    }
}

```

```

if (strcmp(root->right->account_name, key) > 0) {
    root->right->left = splay(root->right->left, key);
    if (root->right->left != NULL) {
        root->right = rightRotate(root->right);
    }
} else if (strcmp(root->right->account_name, key) < 0) {
    root->right->right = splay(root->right->right, key);
    root = leftRotate(root);
}

```

```

return (root->right == NULL) ? root : leftRotate(root);
}
}

```

```

Account* insert(Account* root, char account_name[], char username[], char password[]) {
    if (root == NULL) {
        Account* newNode = (Account*)malloc(sizeof(Account));

```

```

strcpy(newNode->account_name, account_name);
strcpy(newNode->username, username);
strcpy(newNode->password, password);
newNode->left = newNode->right = NULL;
return newNode;
}

root = splay(root, account_name);

int compareResult = strcmp(account_name, root->account_name);
if (compareResult < 0) {
    Account* newNode = (Account*)malloc(sizeof(Account));
    strcpy(newNode->account_name, account_name);
    strcpy(newNode->username, username);
    strcpy(newNode->password, password);
    newNode->left = newNode->right = NULL;
    newNode->left = root->left;
    root->left = NULL;
    newNode->right = root;
    root = newNode;
} else if (compareResult > 0) {
    Account* newNode = (Account*)malloc(sizeof(Account));
    strcpy(newNode->account_name, account_name);
    strcpy(newNode->username, username);
    strcpy(newNode->password, password);
    newNode->left = newNode->right = NULL;
    newNode->right = root->right;
    root->right = NULL;

```



```
    newNode->left = root;
    root = newNode;
}
```

```
return root;
}
```

```
Account* search(Account* root, char account_name[]) {
    return splay(root, account_name);
}
```

```
Account* delete(Account* root, char account_name[]) {
    if (root == NULL) {
        return root;
    }
```

```
    root = splay(root, account_name);
```

```
    if (strcmp(root->account_name, account_name) != 0) {
        return root;
    }
```

```
    Account* temp;
    if (root->left == NULL) {
        temp = root;
        root = root->right;
    } else {
        temp = root;
        root = splay(root->left, account_name);
```

```

    root->right = temp->right;
}

free(temp);
return root;
}

void printInOrder(Account* root) {
    if (root != NULL) {
        printInOrder(root->left);
        printf("Account: %s, Username: %s, Password: %s\n", root->account_name, root->username, root->password);
        printInOrder(root->right);
    }
}

```

Output test cases:

// Test case 1: Add a new account

```
addPassword(&root, "Shopping", "shopper123", "secureShoppingPwd");
```

// Test case 2: Retrieve a password

```
char* retrievedPassword = retrievePassword(root, "Shopping");
```

```
printf("Retrieved Password for Shopping: %s\n", retrievedPassword);
```

// Test case 3: Update an existing account

```
updatePassword(&root, "Shopping", "shopper123", "newSecurePwd123");
```

// Test case 4: Delete an account

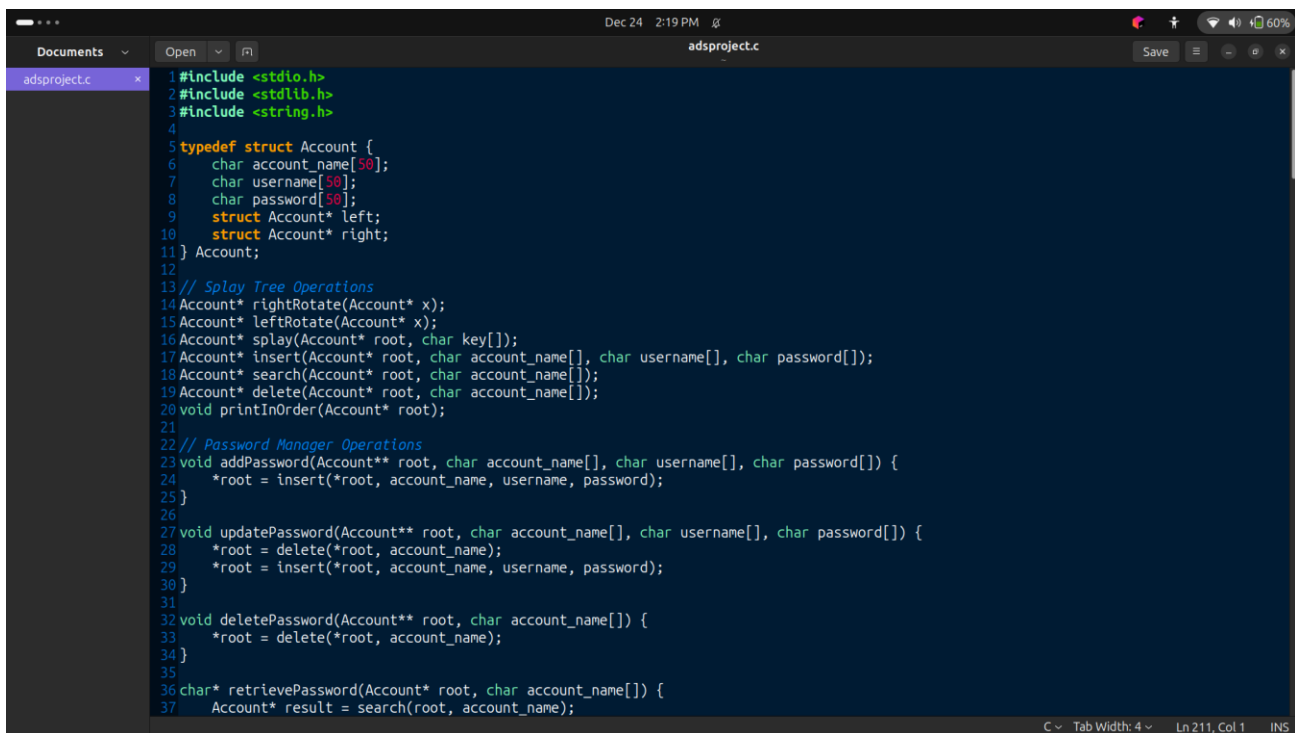
```
deletePassword(&root, "Shopping");
```

// Test case 5: Print the Splay Tree after operations

```
printf("Splay Tree in order:\n");
```

```
printInOrder(root);
```

Screenshot of code and executed output:



The screenshot shows a code editor window titled 'adsproject.c' with the following C code:

```
1#include <stdio.h>
2#include <stdlib.h>
3#include <string.h>
4
5typedef struct Account {
6    char account_name[50];
7    char username[50];
8    char password[50];
9    struct Account* left;
10   struct Account* right;
11} Account;
12
13// Splay Tree Operations
14Account* rightRotate(Account* x);
15Account* leftRotate(Account* x);
16Account* splay(Account* root, char key[]);
17Account* insert(Account* root, char account_name[], char username[], char password[]);
18Account* search(Account* root, char account_name[]);
19Account* delete(Account* root, char account_name[]);
20void printInOrder(Account* root);
21
22// Password Manager Operations
23void addPassword(Account** root, char account_name[], char username[], char password[]) {
24    *root = insert(*root, account_name, username, password);
25}
26
27void updatePassword(Account** root, char account_name[], char username[], char password[]) {
28    *root = delete(*root, account_name);
29    *root = insert(*root, account_name, username, password);
30}
31
32void deletePassword(Account** root, char account_name[]) {
33    *root = delete(*root, account_name);
34}
35
36char* retrievePassword(Account* root, char account_name[]) {
37    Account* result = search(root, account_name);
```

```
Documents  Open  adsproject.c  Save  60%
adsproject.c  x
37 Account* result = search(root, account_name);
38 if (result != NULL) {
39     return result->password;
40 } else {
41     return NULL;
42 }
43 }
44
45 int main() {
46     Account* root = NULL;
47
48     // Test cases
49     addPassword(&root, "Email", "user@example.com", "securePassword1");
50     addPassword(&root, "Bank", "john_doe", "strongPassword123");
51     addPassword(&root, "Social Media", "user123", "password567");
52
53     // Print the Splay Tree in order
54     printf("Splay Tree in order:\n");
55     printInOrder(root);
56
57     // Retrieve a password
58     char* retrievedPassword = retrievePassword(root, "Bank");
59     if (retrievedPassword != NULL) {
60         printf("\nRetrieved Password for Bank: %s\n", retrievedPassword);
61     } else {
62         printf("\nAccount not found\n");
63     }
64
65     // Update a password
66     updatePassword(&root, "Bank", "john_doe", "newStrongPassword456");
67
68     // Print the updated Splay Tree
69     printf("\nSplay Tree after updating Bank account:\n");
70     printInOrder(root);
71
72     // Delete an account
73     deletePassword(&root, "Email");
```

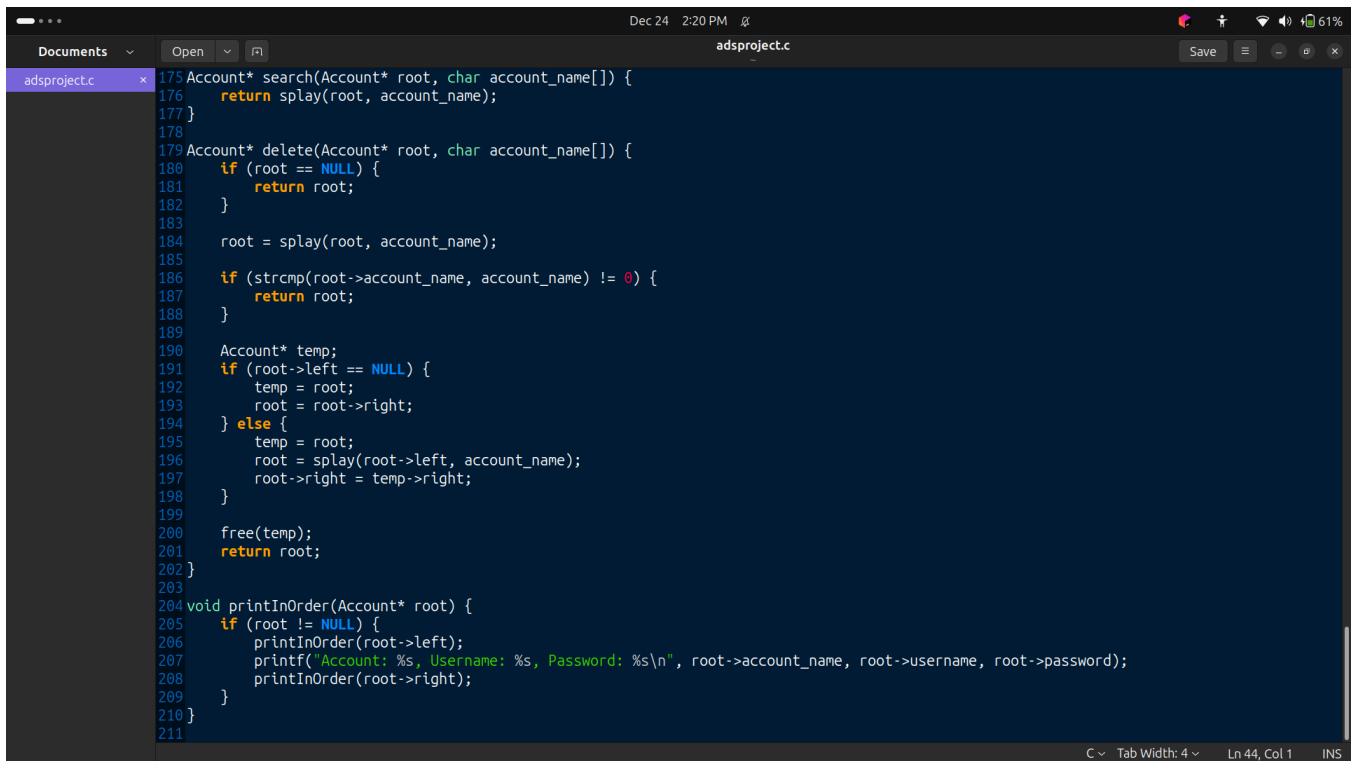
```
Documents  Open  adsproject.c  Save  60%
adsproject.c  x
70     printInOrder(root);
71
72     // Delete an account
73     deletePassword(&root, "Email");
74
75     // Print the Splay Tree after deletion
76     printf("\nSplay Tree after deleting Email account:\n");
77     printInOrder(root);
78
79     return 0;
80 }
81
82 // Splay Tree Implementation
83 Account* rightRotate(Account* x) {
84     Account* y = x->left;
85     x->left = y->right;
86     y->right = x;
87     return y;
88 }
89
90 Account* leftRotate(Account* x) {
91     Account* y = x->right;
92     x->right = y->left;
93     y->left = x;
94     return y;
95 }
96
97 Account* splay(Account* root, char key[]) {
98     if (root == NULL || strcmp(root->account_name, key) == 0) {
99         return root;
100     }
101
102     if (strcmp(root->account_name, key) > 0) {
103         if (root->left == NULL) {
104             return root;
105         }
106         return splay(root->left, key);
107     }
108     if (strcmp(root->account_name, key) < 0) {
109         if (root->right == NULL) {
110             return root;
111         }
112         return splay(root->right, key);
113     }
114     // Root is the parent of the node to be splayed
115     // First, rotate the child to the other side of the root
116     if (strcmp(root->account_name, key) > 0) {
117         root = rightRotate(root);
118     } else {
119         root = leftRotate(root);
120     }
121     // Then, rotate the root with the child
122     if (strcmp(root->account_name, key) > 0) {
123         root = leftRotate(root);
124     } else {
125         root = rightRotate(root);
126     }
127     return root;
128 }
```

```
Dec 24 2:20 PM
Documents
adsproject.c
106
107     if (strcmp(root->left->account_name, key) > 0) {
108         root->left->left = splay(root->left->left, key);
109         root = rightRotate(root);
110     } else if (strcmp(root->left->account_name, key) < 0) {
111         root->left->right = splay(root->left->right, key);
112         if (root->left->right != NULL) {
113             root->left = leftRotate(root->left);
114         }
115     }
116
117     return (root->left == NULL) ? root : rightRotate(root);
118 } else {
119     if (root->right == NULL) {
120         return root;
121     }
122
123     if (strcmp(root->right->account_name, key) > 0) {
124         root->right->left = splay(root->right->left, key);
125         if (root->right->left != NULL) {
126             root->right = rightRotate(root->right);
127         }
128     } else if (strcmp(root->right->account_name, key) < 0) {
129         root->right->right = splay(root->right->right, key);
130         root = leftRotate(root);
131     }
132
133     return (root->right == NULL) ? root : leftRotate(root);
134 }
135 }
136
137 Account* insert(Account* root, char account_name[], char username[], char password[]) {
138     if (root == NULL) {
139         Account* newNode = (Account*)malloc(sizeof(Account));
140         strcpy(newNode->account_name, account_name);
141         strcpy(newNode->username, username);
142         strcpy(newNode->password, password);
143     }
144 }
```

C Tab Width: 4 Ln 44, Col 1 INS

```
Dec 24 2:20 PM
Documents
adsproject.c
140     strcpy(newNode->account_name, account_name);
141     strcpy(newNode->username, username);
142     strcpy(newNode->password, password);
143     newNode->left = newNode->right = NULL;
144     return newNode;
145 }
146
147 root = splay(root, account_name);
148
149 int compareResult = strcmp(account_name, root->account_name);
150 if (compareResult < 0) {
151     Account* newNode = (Account*)malloc(sizeof(Account));
152     strcpy(newNode->account_name, account_name);
153     strcpy(newNode->username, username);
154     strcpy(newNode->password, password);
155     newNode->left = newNode->right = NULL;
156     newNode->left = root->left;
157     root->left = NULL;
158     newNode->right = root;
159     root = newNode;
160 } else if (compareResult > 0) {
161     Account* newNode = (Account*)malloc(sizeof(Account));
162     strcpy(newNode->account_name, account_name);
163     strcpy(newNode->username, username);
164     strcpy(newNode->password, password);
165     newNode->left = newNode->right = NULL;
166     newNode->right = root->right;
167     root->right = NULL;
168     newNode->left = root;
169     root = newNode;
170 }
171
172 return root;
173 }
174
175 Account* search(Account* root, char account_name[]) {
176     return splay(root, account_name);
177 }
```

C Tab Width: 4 Ln 44, Col 1 INS



```
175 Account* search(Account* root, char account_name[]) {
176     return splay(root, account_name);
177 }
178
179 Account* delete(Account* root, char account_name[]) {
180     if (root == NULL) {
181         return root;
182     }
183
184     root = splay(root, account_name);
185
186     if (strcmp(root->account_name, account_name) != 0) {
187         return root;
188     }
189
190     Account* temp;
191     if (root->left == NULL) {
192         temp = root;
193         root = root->right;
194     } else {
195         temp = root;
196         root = splay(root->left, account_name);
197         root->right = temp->right;
198     }
199
200     free(temp);
201     return root;
202 }
203
204 void printInOrder(Account* root) {
205     if (root != NULL) {
206         printInOrder(root->left);
207         printf("Account: %s, Username: %s, Password: %s\n", root->account_name, root->username, root->password);
208         printInOrder(root->right);
209     }
210 }
211
```

Output:

```
aimer@aimer-Inspiron-15-7000-Gaming:~$ gedit adsproject.c
aimer@aimer-Inspiron-15-7000-Gaming:~$ gcc adsproject.c
aimer@aimer-Inspiron-15-7000-Gaming:~$ ./a.out
Splay Tree in order:
Account: Bank, Username: john_doe, Password: strongPassword123
Account: Email, Username: user@example.com, Password: securePassword1
Account: Social Media, Username: user123, Password: password567

Retrieved Password for Bank: strongPassword123

Splay Tree after updating Bank account:
Account: Bank, Username: john_doe, Password: newStrongPassword456
Account: Social Media, Username: user123, Password: password567

Splay Tree after deleting Email account:
Account: Bank, Username: john_doe, Password: newStrongPassword456
Account: Social Media, Username: user123, Password: password567
aimer@aimer-Inspiron-15-7000-Gaming:~$
```