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

//Membuat struct tree
struct tree{
  int num;
  int award;
  char name[55];
  struct tree *left, *right;
}*root = NULL;

//Struct untuk membuat root
struct tree *createNode(int num, const char *name, int award) {
  struct tree *newNode = (struct tree*)malloc(sizeof(struct tree));
  newNode->num = num;
  strcpy(newNode->name,name);
  newNode->award = award;
  newNode->left = newNode->right = NULL;
  return newNode;
}

//Struct untuk menambah node
struct tree *insertNode(struct tree *root, int num, const char *name, int award, int height){
    char pos[55];
    if(height < 3){
        if(!root){
            return createNode(num, name, award);
        }
    else{
    while(height < 3){
        if(num < root->num){
```

```
            root->left = insertNode(root->left, num, name, award, height+1);
        }
        else if(num > root->num){
            root->right = insertNode(root->right, num, name, award, height+1);
        }
        break;
        }
    }
    }
    else if(height == 3){
        puts("--- Maximum Tree Level is 3 ---");
    }
}

//Struct untuk mencari data
struct tree *search(struct tree *root, int num){
 if(root != NULL){
  if(root->num == num) {
  return root;
  }
  else{
   struct tree *temp;
   temp = search(root->left, num);
   if(temp == NULL){
   temp = search(root->right, num);
   }
   return temp;
  }
 }
}
```

```
//Struct untuk delete data
struct tree *successor(struct tree *root) {
   struct tree *temp = root;
   while (temp && temp->right != NULL)
   temp = temp->right;
   return temp;
}

struct tree *delData(struct tree *root, int num){
   if(root == NULL) {
      return root;
   }
   if(num < root->num){
      root->left = delData(root->left, num);
   }
   else if (num > root->num){
      root->right = delData(root->right, num);
   }
   else{
      if (root->left == NULL){
         struct tree *temp = root->right;
         printf("S%d - %s  <%d> is deleted\n", root->num, root->name, root->award);
         printf("--- Delete data success ---\n\n");
         free(root);
         return temp;
      }
      else if (root->right == NULL){
         struct tree *temp = root->left;
         printf("S%d - %s  <%d> is delete\n", root->num, root->name, root->award);
         printf("--- Delete data success ---\n\n");
         free(root);
```

```
            return temp;
        }
        else{
            struct tree *temp = successor(root->right);
            root->num = temp->num;
            root->right = delData(root->right, temp->num);
            printf("S%d - %s  <%d> is deleted.\n", root->num, root->name, root->award);
            printf("--- Delete data success ---\n\n");
        }
    }
    return root;
}

//Fungsi untuk update data
void update(struct tree *root, const char *newName,int newAward){
    if(root!=NULL){
        strcpy(root->name, newName);
        root->award = newAward;
    }
}

//Fungsi PreOrder, InOrder, PostOrder
void PreOrder(struct tree *root){
    if(root != NULL){
        printf("- %d     %s   [  %d]\n\n", root->num, root->name, root->award);
        PreOrder(root->left);
        PreOrder(root->right);
    }
}

void InOrder(struct tree *root){
```

```
    if(root != NULL){
        InOrder(root->left);
        printf("- %d     %s   [  %d]\n\n", root->num, root->name, root->award);
        InOrder(root->right);
    }
}

void PostOrder(struct tree *root){
    if(root!= NULL){
        PostOrder(root->left);
        PostOrder(root->right);
        printf("- %d     %s   [  %d]\n\n", root->num, root->name, root->award);
    }
}

//Fungsi untuk menampilkan menu
void display(){
    printf("Internship Employee\n");
    printf("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n\n");
    printf("1. Add a new data\n");
    printf("2. Update a certain data\n");
    printf("3. InOrder, PreOrder, PostOrder by internship employee ID\n");
    printf("4. Delete a certain data\n");
    printf("5. Exit\n\n");
}

//Fungsi utama
int main(){
    int menu, num, anoNum, award, newAward, height;
    char name[55], newName[55];
    display();
```

```
printf(">> Input choice : ");
scanf("%d",&menu);
if(menu == 1){
   printf("\nInput internship employee ID S[1-9][0-9]: S");
   scanf("%d",&num);
   printf("Input internship employee name: ");
   scanf("%s",name);
   printf("Input total awards received [1..100]: ");
   scanf("%d",&award);
   if(search(root, num) == NULL){
      root = insertNode(root, num, name, award, height);
      printf("\n--- Add New Data Success ---\n\n");
      }
   else if(search(root,num) != NULL){
      printf("\n--- Data Already Exist ---\n\n");
   }
   return main();
}
else if(menu == 2){
   if(root == NULL){
      printf("\n--- There is no data in the Tree ---\n\n");
   }
   else{
      printf("\nInput internship employee ID S[1-9][0-9]: S");
      scanf("%d",&num);
      if(search(root, num) == NULL){
         printf("\n--- Internship Employee ID is not Found ---\n\n");
      }
      else if(search(root, num) != NULL){
         printf("\nIntership Employee Id   : S%d\n",root->num);
```

```
            printf("Intership Employee name : %s\n",root->name);
            printf("Total awards received    : %d\n",root->award);
            printf("\nInput internship employee name: ");
            scanf("%s",&newName);
            printf("\nInput total awards recceived [1..100]: ");
            scanf("%d",&newAward);
            printf("\n--- Update data success ---\n\n");
            update(root, newName, newAward);
        }
    }
return main();
}
else if(menu == 3){
    if(root == NULL){
        printf("\n--- There is no data in the Tree ---\n\n");
    }
    else{
        printf("\nPreOrder  :");
        PreOrder(root);
        printf("\nInOrder  :");
        InOrder(root);
        printf("\nPostOrder  :");
        PostOrder(root);
    }
return main();
}
else if(menu == 4){
    if(root == NULL){
        printf("\n--- There is no data in the Tree ---\n\n");
    }
    else{
```

```
            printf("\nInput internship employee ID S[1-9][0-9]: S");
            scanf("%d",&num);
            if(search(root, num) == NULL){
               printf("\n--- Internship Employee ID is not Found ---\n\n");
            }
            else if(search(root, num) != NULL){
               root = delData(root, num);
            }
         }
         return main();
      }
      else if(menu == 5){
         exit(0);
      }
      else{
         printf("\n--- Wrong Menu ---\n\n");
      }
   }
```