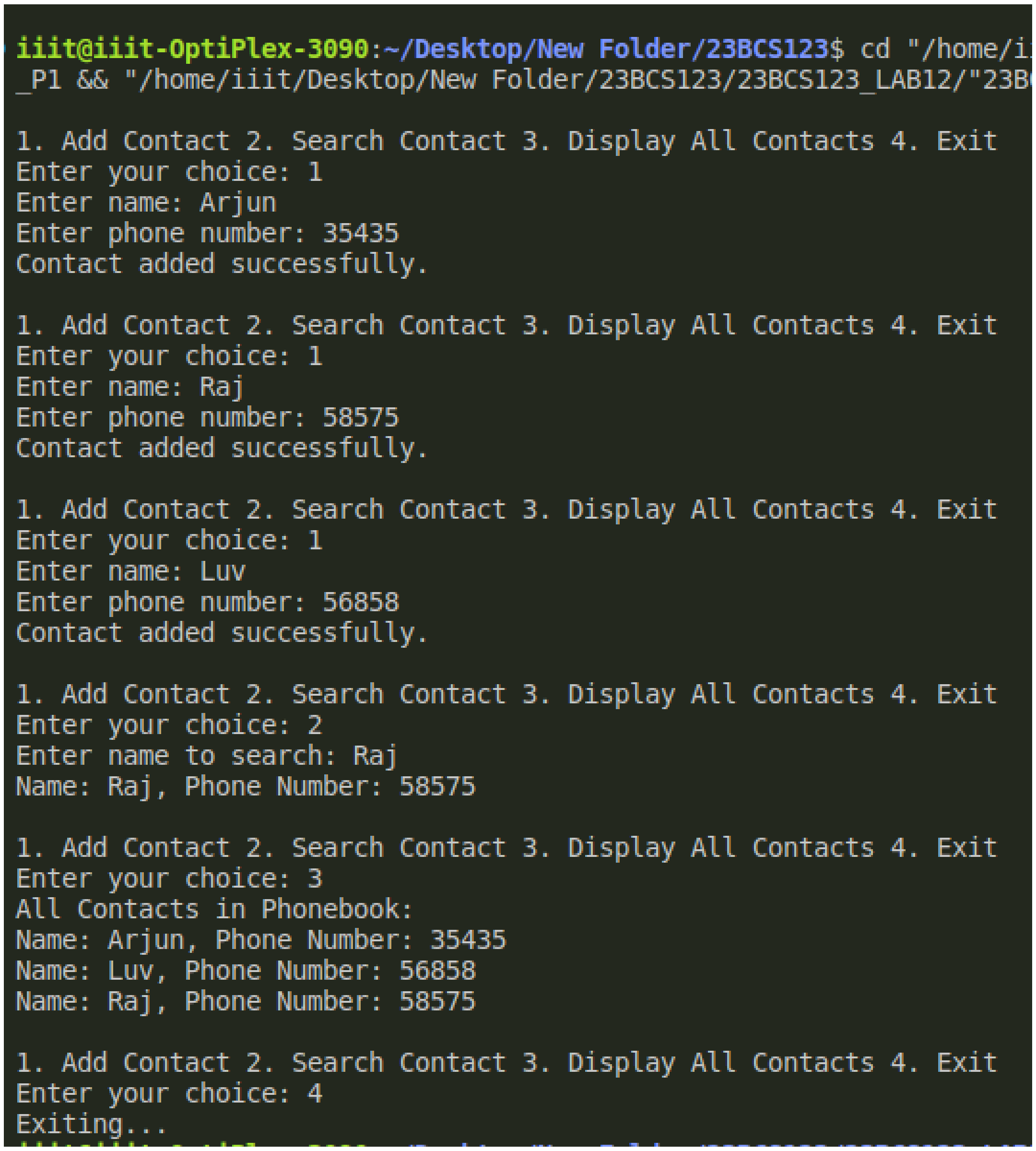
CS102 - Lab 12 - 04/04/2024

```c
//SHIVAMBU DEV PANDEY
//23BCS123
//CSE B
//LAB 12
//PROGRAM 1
//Date : 04/04/2024

/*
1.Implement a phonebook application using a Binary Search Tree in C.Each
entry should contain a name and corresponding phone number. Provided the
name, the program should display the contact details of the matching name.
Also, display the contact details of all the names in the BST in in-order
manner.
*/

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

typedef struct PhonebookEntry {
   char name[50];
   char number[15];
   struct PhonebookEntry* left;
   struct PhonebookEntry* right;
} PBE;

PBE* create(const char* name, const char* number) {
   PBE* contact = (PBE*)malloc(sizeof(PBE));
   if (contact == NULL) {
       printf("Memory allocation failed.\n");
       exit(1);
   }
   strcpy(contact->name, name);
   strcpy(contact->number, number);
   contact->left = NULL;
   contact->right = NULL;
   return contact;
}
```

```c
PBE* insert(PBE* root, const char* name, const char* number) {
   if (root == NULL) {
       root = create(name, number);
   } else {
       if (strcmp(name, root->name) < 0)
           root->left = insert(root->left, name, number);
       else if (strcmp(name, root->name) > 0)
           root->right = insert(root->right, name, number);
       else
           printf("Contact with name '%s' already exists.\n", name);
   }
   return root;
}

PBE* search(PBE* root, const char* name) {
   if (root == NULL || strcmp(root->name, name) == 0)
       return root;
   if (strcmp(name, root->name) < 0)
       return search(root->left, name);
   return search(root->right, name);
}

void display(PBE* root) {
   if (root != NULL) {
       display(root->left);
       printf("Name: %s, Phone Number: %s\n", root->name, root->number);
       display(root->right);
   }
}

int main() {
   PBE* phonebook = NULL;
   int choice;
   char name[50], number[15];

   while (1) {
       printf("\n1. Add Contact 2. Search Contact 3. Display All Contacts
4. Exit\n");
       printf("Enter your choice: ");
```

```
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter name: ");
                scanf("%s", name);
                printf("Enter phone number: ");
                scanf("%s", number);
                phonebook = insert(phonebook, name, number);
                printf("Contact added successfully.\n");
                break;
            case 2:
                printf("Enter name to search: ");
                scanf("%s", name);
                PBE* result = search(phonebook, name);
                if (result != NULL)
                    printf("Name: %s, Phone Number: %s\n", result->name,
result->number);
                else
                    printf("Contact with name '%s' not found.\n", name);
                break;
            case 3:
                printf("All Contacts in Phonebook:\n");
                display(phonebook);
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice.\n");
        }
    }

    return 0;
}
```

OUTPUT 1 :



```
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123$ cd "/home/i
_P1 && "/home/iiit/Desktop/New Folder/23BCS123/23BCS123_LAB12/"23B

1. Add Contact 2. Search Contact 3. Display All Contacts 4. Exit
Enter your choice: 1
Enter name: Arjun
Enter phone number: 35435
Contact added successfully.

1. Add Contact 2. Search Contact 3. Display All Contacts 4. Exit
Enter your choice: 1
Enter name: Raj
Enter phone number: 58575
Contact added successfully.

1. Add Contact 2. Search Contact 3. Display All Contacts 4. Exit
Enter your choice: 1
Enter name: Luv
Enter phone number: 56858
Contact added successfully.

1. Add Contact 2. Search Contact 3. Display All Contacts 4. Exit
Enter your choice: 2
Enter name to search: Raj
Name: Raj, Phone Number: 58575

1. Add Contact 2. Search Contact 3. Display All Contacts 4. Exit
Enter your choice: 3
All Contacts in Phonebook:
Name: Arjun, Phone Number: 35435
Name: Luv, Phone Number: 56858
Name: Raj, Phone Number: 58575

1. Add Contact 2. Search Contact 3. Display All Contacts 4. Exit
Enter your choice: 4
Exiting...
```

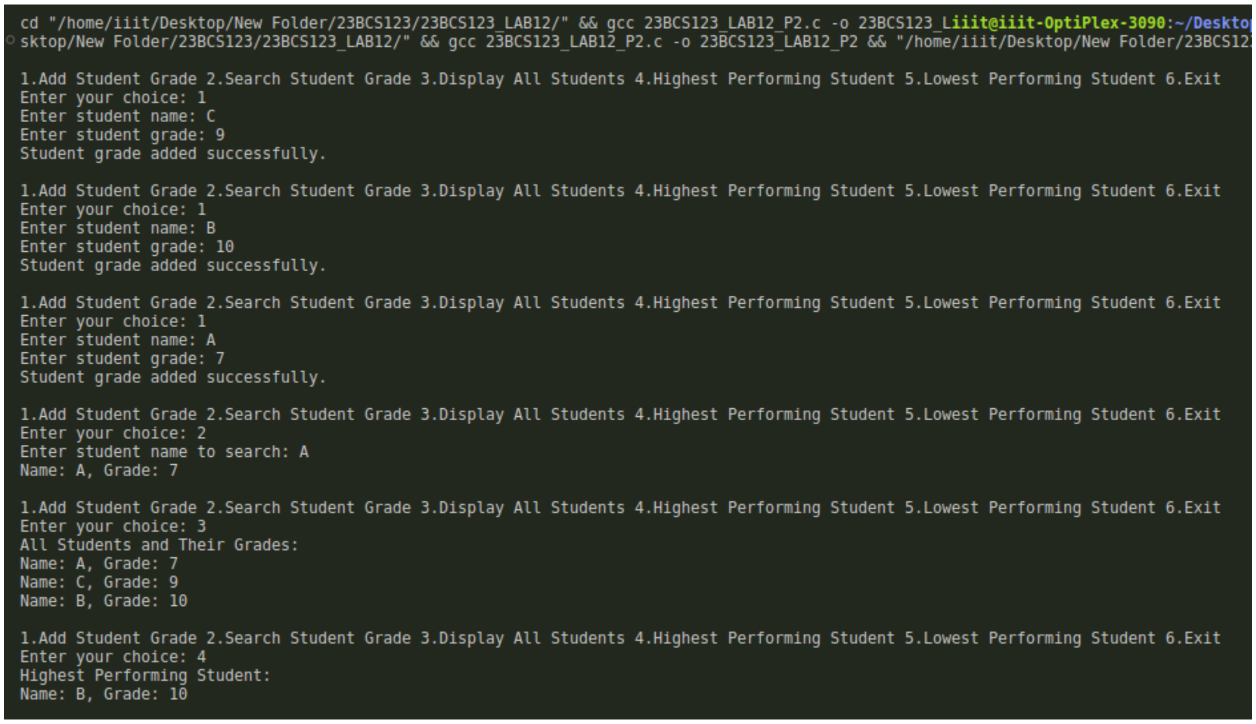
```c
//SHIVAMBU DEV PANDEY
//23BCS123
//CSE B
//LAB 12
//PROGRAM 2
//Date : 04/04/2024

/*
2.Design a student grade tracking system using a Binary Search Tree in C.
Each node represents a student, with associated grades. The program should
be able search for a particular student and print the grade of that
student.
Also, display all the student (in-order) details and identify the highest
and lowest performing students from the BST.
*/

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

typedef struct Student {
   char name[50];
   int grade;
   struct Student* left;
   struct Student* right;
} Student;

Student* create(const char* name, int grade) {
   Student* newStudent = (Student*)malloc(sizeof(Student));
   if (newStudent == NULL) {
       printf("Memory allocation failed.\n");
       exit(1);
   }
   strcpy(newStudent->name, name);
   newStudent->grade = grade;
   newStudent->left = NULL;
   newStudent->right = NULL;
   return newStudent;
}
```

```c
Student* insert(Student* root, const char* name, int grade) {
   if (root == NULL) {
       root = create(name, grade);
   } else {
       if (grade < root->grade)
           root->left = insert(root->left, name, grade);
       else if (grade > root->grade)
           root->right = insert(root->right, name, grade);
       else
           printf("Student with name '%s' and grade '%d' already
exists.\n", name, grade);
   }
   return root;
}

Student* search(Student* root, const char* name) {
   if (root == NULL || strcmp(root->name, name) == 0)
       return root;
   else {
       Student* left_result = search(root->left, name);
       if (left_result != NULL)
           return left_result;
       else
           return search(root->right, name);
   }
}

void display(Student* root) {
   if (root != NULL) {
       display(root->left);
       printf("Name: %s, Grade: %d\n", root->name, root->grade);
       display(root->right);
   }
}

Student* findHighest(Student* root) {
   if (root == NULL)
       return NULL;
   while (root->right != NULL)
       root = root->right;
```

```
    return root;
}

Student* findLowest(Student* root) {
    if (root == NULL)
        return NULL;
    while (root->left != NULL)
        root = root->left;
    return root;
}

int main() {
    Student* stud = NULL;
    int choice, grade;
    char name[50];

    while (1) {
        printf("\n1.Add Student Grade 2.Search Student Grade 3.Display All
Students 4.Highest Performing Student 5.Lowest Performing Student
6.Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter student name: ");
                scanf("%s", name);
                printf("Enter student grade: ");
                scanf("%d", &grade);
                stud = insert(stud, name, grade);
                printf("Student grade added successfully.\n");
                break;
            case 2:
                printf("Enter student name to search: ");
                scanf("%s", name);
                Student* result = search(stud, name);
                if (result != NULL)
                    printf("Name: %s, Grade: %d\n", result->name,
result->grade);
                else
```

```
                    printf("Student with name '%s' not found.\n", name);
                break;

            case 3:
                printf("All Students and Their Grades:\n");
                display(stud);
                break;
            case 4:
                printf("Highest Performing Student:\n");
                Student* highest = findHighest(stud);
                if (highest != NULL)
                    printf("Name: %s, Grade: %d\n", highest->name,
highest->grade);
                else
                    printf("No students found.\n");
                break;
            case 5:
                printf("Lowest Performing Student:\n");
                Student* lowest = findLowest(stud);
                if (lowest != NULL)
                    printf("Name: %s, Grade: %d\n", lowest->name,
lowest->grade);
                else
                    printf("No students found.\n");
                break;
            case 6:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice.\n");
        }
    }

    return 0;
}
```

OUTPUT:

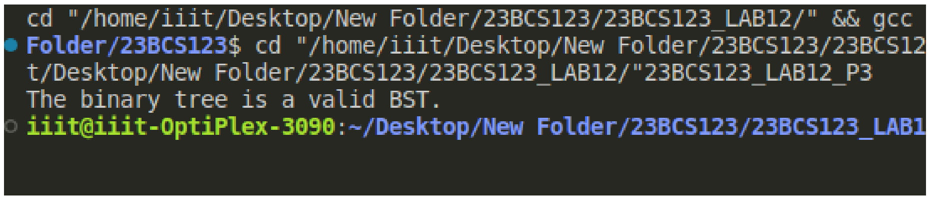


```
cd "/home/iiit/Desktop/New Folder/23BCS123/23BCS123_LAB12/" && gcc 23BCS123_LAB12_P2.c -o 23BCS123_Liiit@iiit-OptiPlex-3090:~/Deskto
sktop/New Folder/23BCS123/23BCS123_LAB12/" && gcc 23BCS123_LAB12_P2.c -o 23BCS123_LAB12_P2 && "/home/iiit/Desktop/New Folder/23BCS12

1.Add Student Grade 2.Search Student Grade 3.Display All Students 4.Highest Performing Student 5.Lowest Performing Student 6.Exit
Enter your choice: 1
Enter student name: C
Enter student grade: 9
Student grade added successfully.

1.Add Student Grade 2.Search Student Grade 3.Display All Students 4.Highest Performing Student 5.Lowest Performing Student 6.Exit
Enter your choice: 1
Enter student name: B
Enter student grade: 10
Student grade added successfully.

1.Add Student Grade 2.Search Student Grade 3.Display All Students 4.Highest Performing Student 5.Lowest Performing Student 6.Exit
Enter your choice: 1
Enter student name: A
Enter student grade: 7
Student grade added successfully.

1.Add Student Grade 2.Search Student Grade 3.Display All Students 4.Highest Performing Student 5.Lowest Performing Student 6.Exit
Enter your choice: 2
Enter student name to search: A
Name: A, Grade: 7

1.Add Student Grade 2.Search Student Grade 3.Display All Students 4.Highest Performing Student 5.Lowest Performing Student 6.Exit
Enter your choice: 3
All Students and Their Grades:
Name: A, Grade: 7
Name: C, Grade: 9
Name: B, Grade: 10

1.Add Student Grade 2.Search Student Grade 3.Display All Students 4.Highest Performing Student 5.Lowest Performing Student 6.Exit
Enter your choice: 4
Highest Performing Student:
Name: B, Grade: 10
```

```c
//SHIVAMBU DEV PANDEY
//23BCS123
//CSE B
//LAB 12
//PROGRAM 3
//Date : 04/04/2024

/*
3.Write a C program to check if a given binary tree is a valid
binary search tree, where the values of nodes in the left subtree
are less than the node value and values in the right subtree are greater.
*/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>

typedef struct node {
   int data;
   struct node* left;
   struct node* right;
}node;

node* create(int val) {
   node* newNode = (node*)malloc(sizeof(node));
   if (newNode == NULL) {
       printf("Memory allocation failed.\n");
       exit(1);
   }
   newNode->data = val;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}

bool checkBST(node* root, int min, int max) {
   if (root == NULL)
       return true;
```

```c
    if (root->data < min || root->data > max)
        return false;

    return checkBST(root->left, min, root->data - 1) &&
checkBST(root->right, root->data + 1, max);
}

int main() {

    node* root = create(50);
    root->left = create(17);
    root->right = create(72);
    root->left->left = create(12);
    root->left->right = create(23);
    root->left->left->left = create(9);
    root->left->left->right = create(14);
    root->left->right->left = create(19);
    root->right->left = create(54);
    root->right->right = create(76);

    root->right->left->right = create(67);

    if (checkBST(root, INT_MIN, INT_MAX))
        printf("The binary tree is a valid BST.\n");
    else
        printf("The binary tree is not a valid BST.\n");

    return 0;
}
```

OUTPUT 3 :



```
cd "/home/iiit/Desktop/New Folder/23BCS123/23BCS123_LAB12/" && gcc
Folder/23BCS123$ cd "/home/iiit/Desktop/New Folder/23BCS123/23BCS12
t/Desktop/New Folder/23BCS123/23BCS123_LAB12/"23BCS123_LAB12_P3
The binary tree is a valid BST.
iiit@iiit-OptiPlex-3090:~/Desktop/New Folder/23BCS123/23BCS123_LAB1
```