1. Creation Of Tree

```c
#include<stdio.h>
#include<stdlib.h>
struct tree {
    int val;
    struct tree *left,*right;
};
struct tree* create() {
    struct tree* node;
    node=(struct tree*)malloc(sizeof(struct tree));
    int x;
    printf("Enter data(-1 for exit)=");scanf("%d",&node->val);
    x=node->val;
    if(x==-1) return 0;
    printf("Enter left child of %d\n",x);
    node->left=create();
    printf("Enter right child of %d\n",x);
    node->right=create();
    return node;
}
void printTree(struct tree* root) {
    if(root==NULL) return;
    printTree(root->left);
    printf("%d ",root->val);
    printTree(root->right);
}
int main() {
    struct tree *root= NULL;
    root=create();
    printTree(root);
}
Output:
```

```
Enter data(-1 for exit)=5
Enter left child of 5
Enter data(-1 for exit)=4
Enter left child of 4
Enter data(-1 for exit)=2
Enter left child of 2
Enter data(-1 for exit)=-1
Enter right child of 2
Enter data(-1 for exit)=-1
Enter right child of 4
Enter data(-1 for exit)=1
Enter left child of 1
Enter data(-1 for exit)=-1
Enter right child of 1
Enter data(-1 for exit)=-1
Enter right child of 5
Enter data(-1 for exit)=3
Enter left child of 3
Enter data(-1 for exit)=-1
Enter right child of 3
Enter data(-1 for exit)=2
Enter left child of 2
Enter data(-1 for exit)=-1
Enter right child of 2
Enter data(-1 for exit)=-1
2 4 1 5 3 2
```

2. Tree Traversals

```c
#include<stdio.h>
#include<stdlib.h>
struct tree {
    int val;
    struct tree *left,*right;
```

```
};
struct tree* create() {
    struct tree* node;
    node=(struct tree*)malloc(sizeof(struct tree));
    int x;
    printf("Enter data(-1 for exit)=");scanf("%d",&node->val);
    x=node->val;
    if(x==-1) return 0;
    printf("Enter left child of %d\n",x);
    node->left=create();
    printf("Enter right child of %d\n",x);
    node->right=create();
    return node;
}
void preorder(struct tree* root) {
    if(root==NULL) return;
    printf("%d ",root->val);
    preorder(root->left);
    preorder(root->right);
}
void inorder(struct tree* root) {
    if(root==NULL) return;
    inorder(root->left);
    printf("%d ",root->val);
    inorder(root->right);
}
void postorder(struct tree* root) {
    if(root==NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ",root->val);
}
int main() {
    struct tree *root= NULL;
    root=create();
    printf("Preorder: ");
    preorder(root);
    printf("\n");
    printf("Inorder: ");
    inorder(root);
```

```
    printf("\n");
    printf("Postorder: ");
    postorder(root);
    printf("\n");
}
```

Output:

:

```
Enter data(-1 for exit)=5
Enter left child of 5
Enter data(-1 for exit)=4
Enter left child of 4
Enter data(-1 for exit)=2
Enter left child of 2
Enter data(-1 for exit)=-1
Enter right child of 2
Enter data(-1 for exit)=-1
Enter right child of 4
Enter data(-1 for exit)=1
Enter left child of 1
Enter data(-1 for exit)=-1
Enter right child of 1
Enter data(-1 for exit)=-1
Enter right child of 5
Enter data(-1 for exit)=3
Enter left child of 3
Enter data(-1 for exit)=-1
Enter right child of 3
Enter data(-1 for exit)=2
Enter left child of 2
Enter data(-1 for exit)=-1
Enter right child of 2
Enter data(-1 for exit)=-1
Preorder: 5 4 2 1 3 2
Inorder: 2 4 1 5 3 2
Postorder: 2 1 4 2 3 5
```

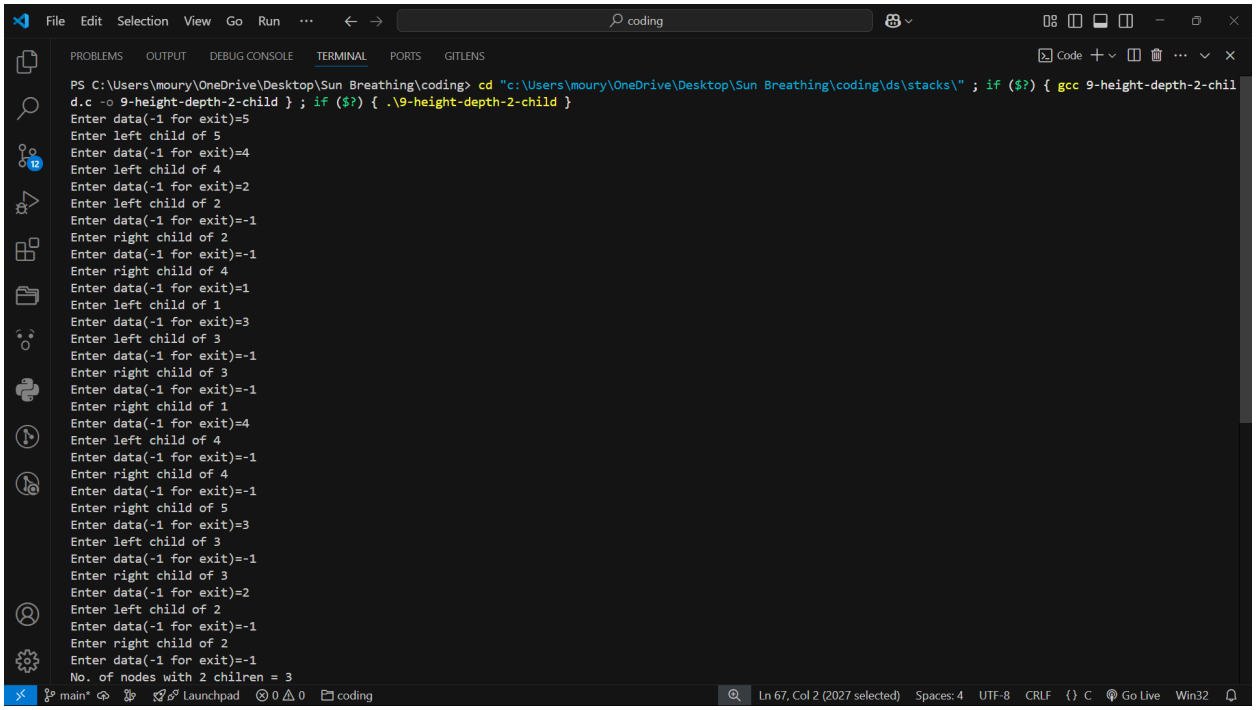
3.Finding height of a tree, depth of a tree, deepest node in the tree, no. of nodes with 2 children

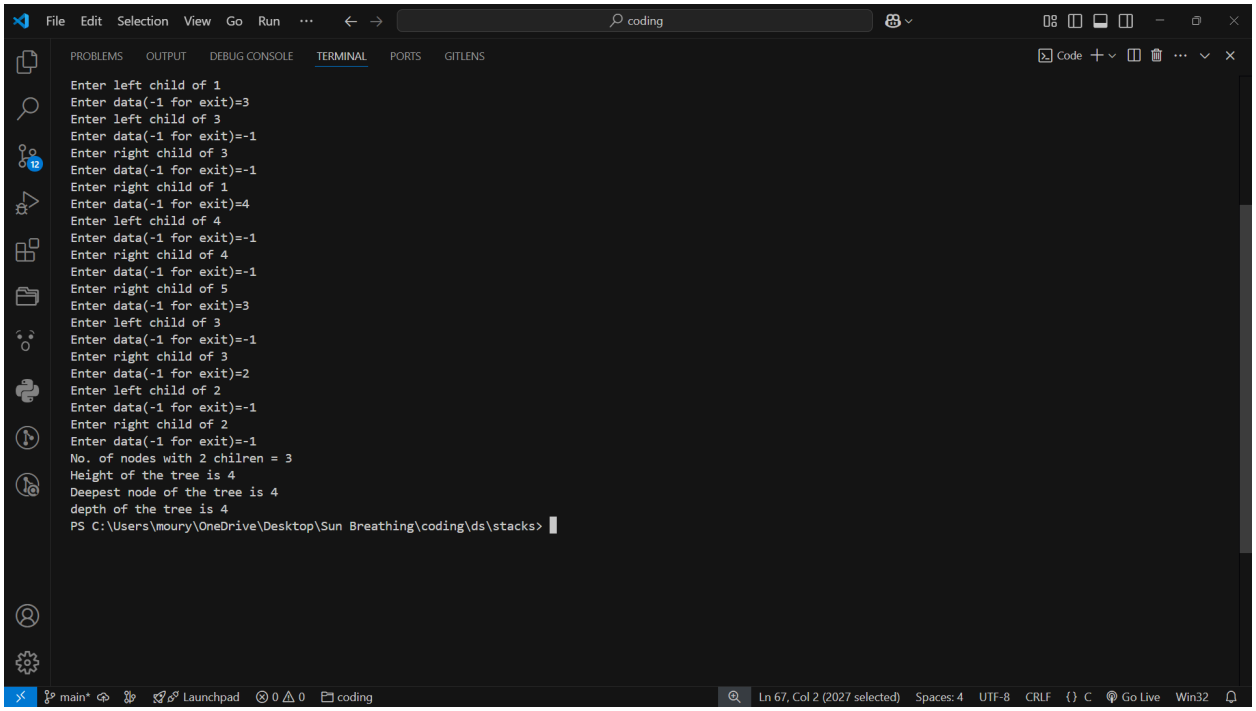
```c
#include<stdio.h>
#include<stdlib.h>
struct tree {
    int val;
```

```
    struct tree *left,*right;
};
struct tree* create() {
    struct tree* node;
    node=(struct tree*)malloc(sizeof(struct tree));
    int x;
    printf("Enter data(-1 for exit)=");scanf("%d",&node->val);
    x=node->val;
    if(x==-1) return 0;
    printf("Enter left child of %d\n",x);
    node->left=create();
    printf("Enter right child of %d\n",x);
    node->right=create();
    return node;
}
int heightoftree(struct tree* root) {
    if (root == NULL) return 0;
    int leftHeight = heightoftree(root->left);
    int rightHeight = heightoftree(root->right);
    if (leftHeight > rightHeight) return leftHeight + 1;
    else return rightHeight + 1;
}

int depthoftree(struct tree* root) {
    if (root == NULL) return 0;
    int leftDepth = depthoftree(root->left);
    int rightDepth = depthoftree(root->right);
    if (leftDepth > rightDepth) return leftDepth + 1;
    else return rightDepth + 1;
}
struct tree* deepestnode(struct tree* root) {
    if (root == NULL) return NULL;
    struct tree* temp = NULL;
    struct tree** queue = (struct tree**)malloc(1000 * sizeof(struct
tree*));
    int front = 0, rear = 0;
    queue[rear++] = root;
    while (front < rear) {
        temp = queue[front++];
        if (temp->left) queue[rear++] = temp->left;
```

```
        if (temp->right) queue[rear++] = temp->right;
    }
    free(queue);
    return temp;
}

int twoChildren(struct tree* root) {
    if(root==NULL) return 0;
    int l=0;
    if(root->left && root->right) {
        l++;
    }
    l+=twoChildren(root->left);
    l+=twoChildren(root->right);
    return l;
}
int main() {
    struct tree *root= NULL;
    root=create();
    printf("No. of nodes with 2 chilren = %d\n",twoChildren(root));
    printf("Height of the tree is %d\n",heightoftree(root));
    printf("Deepest node of the tree is %d\n",deepestnode(root)->val);
    printf("depth of the tree is %d\n",depthoftree(root));
}
```

Output:


File Edit Selection View Go Run
coding
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Code
PS C:\Users\moury\OneDrive\Desktop\Sun Breathing\coding> cd "c:\Users\moury\OneDrive\Desktop\Sun Breathing\coding\ds\stacks\" ; if ($?) { gcc 9-height-depth-2-child.c -o 9-height-depth-2-child } ; if ($?) { .\9-height-depth-2-child }
Enter data(-1 for exit)=5
Enter left child of 5
Enter data(-1 for exit)=4
Enter left child of 4
Enter data(-1 for exit)=2
Enter left child of 2
Enter data(-1 for exit)=-1
Enter right child of 2
Enter data(-1 for exit)=-1
Enter right child of 4
Enter data(-1 for exit)=1
Enter left child of 1
Enter data(-1 for exit)=3
Enter left child of 3
Enter data(-1 for exit)=-1
Enter right child of 3
Enter data(-1 for exit)=-1
Enter right child of 1
Enter data(-1 for exit)=4
Enter left child of 4
Enter data(-1 for exit)=-1
Enter right child of 4
Enter data(-1 for exit)=-1
Enter right child of 5
Enter data(-1 for exit)=3
Enter left child of 3
Enter data(-1 for exit)=-1
Enter right child of 3
Enter data(-1 for exit)=2
Enter left child of 2
Enter data(-1 for exit)=-1
Enter right child of 2
Enter data(-1 for exit)=-1
No. of nodes with 2 chilren = 3
main* Launchpad 0 0 coding
Ln 67, Col 2 (2027 selected) Spaces: 4 UTF-8 CRLF {} C Go Live Win32


File Edit Selection View Go Run
coding
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Code
Enter left child of 1
Enter data(-1 for exit)=3
Enter left child of 3
Enter data(-1 for exit)=-1
Enter right child of 3
Enter data(-1 for exit)=-1
Enter right child of 1
Enter data(-1 for exit)=4
Enter left child of 4
Enter data(-1 for exit)=-1
Enter right child of 4
Enter data(-1 for exit)=-1
Enter right child of 5
Enter data(-1 for exit)=3
Enter left child of 3
Enter data(-1 for exit)=-1
Enter right child of 3
Enter data(-1 for exit)=2
Enter left child of 2
Enter data(-1 for exit)=-1
Enter right child of 2
Enter data(-1 for exit)=-1
No. of nodes with 2 chilren = 3
Height of the tree is 4
Deepest node of the tree is 4
depth of the tree is 4
PS C:\Users\moury\OneDrive\Desktop\Sun Breathing\coding\ds\stacks>
main* Launchpad 0 0 coding
Ln 67, Col 2 (2027 selected) Spaces: 4 UTF-8 CRLF {} C Go Live Win32