```c
#include "stdio.h"
#include "stdlib.h"
#include "malloc.h"

typedef struct tree {
	char key;
	struct tree *left;
	struct tree *right;
} node;

node *create_root(node *root,char key) {
	root = (node*)malloc(sizeof(node));
	root->key = key;
	root->left = root->right = NULL;
	return root;
}

node *add_node(node *root, char key) {
	if (root == 0) {
		return create_root(root, key);
	}
	if (root->key <= key) {
		if (root->right == 0) {
			root->right = create_root(root, key);
			return root;
		}
		add_node(root->right, key);

	}
	else {
		if (root->left == 0) {
			root->left = create_root(root, key);
			return root;
		}
		add_node(root->left, key);
	}
	return root;
}

void print_tree(node *root) {
	static int l = 0;
	l++;
	if (root != NULL) {
		print_tree(root->right);
		for (int i = 0; i < l; i++)
			printf("    ");
		printf("-——%c\n", root->key);
		print_tree(root->left);
	}
	l--;
}

void remove_node(node **root, char key) {
	node *repl = NULL,
		*parent = NULL,
		*tmp = *root;

	while ((tmp != NULL) && (tmp->key != key)) {
		parent = tmp;

		if (key < tmp->key)
			tmp = tmp->left;
		else
			tmp = tmp->right;
	}

	if (tmp == NULL)
		return;
```

```
        if ((tmp->left != NULL) && (tmp->right == NULL)) {
                if (parent != NULL) {
                        if (parent->left == tmp)
                                parent->left = tmp->left;
                        else
                                parent->right = tmp->left;
                }
                else
                        *root = tmp->left;
                free(tmp);
                tmp = NULL;
        }
        else if (tmp->left == NULL && tmp->right != NULL) {
                if (parent != NULL) {
                        if (parent->left == tmp)
                                parent->left = tmp->right;
                        else
                                parent->right = tmp->right;
                }
                else
                        *root = tmp->right;
                free(tmp);
                tmp = NULL;
        }
        else if (tmp->left != NULL && tmp->right != NULL) {
                repl = tmp->right;
                if (repl->left == NULL)
                        tmp->right = repl->right;
                else {
                        while (repl->left != NULL) {
                                parent = repl;
                                repl = repl->left;
                        }
                        parent->left = repl->right;
                }
                tmp->key = repl->key;
                free(repl);
                repl = NULL;
        }
        else {
                if (parent != NULL) {
                        if (parent->left == tmp)
                                parent->left = NULL;
                        else
                                parent->right = NULL;
                }
                else
                        *root = NULL;
                free(tmp);
                tmp = NULL;
        }
        return;
}

int task(node *root, int *k) {
        int p = 0;
        if ((root->right != NULL) || (root->left != NULL)) {
                p++;
                *k = *k + p;
        }
        else
                return ;
        if (root->left != NULL)
                task(root->left, k);
        if (root->right != NULL)
                task(root->right, k);
        return *k;
}
```

```c
void menu() {
	printf("============================\n"
		"|| What you want to do?  ||\n"
		"||     1-add node         ||\n"
		"||     2-print tree       ||\n"
		"||     3-remove node      ||\n"
		"||     4-curry to task    ||\n"
		"||     5-menu             ||\n"
		"||     0-end              ||\n"
		"============================\n");
}

int main() {
	node *a = NULL;
	int ch = 10, x = 0, k;
	char key;
	printf("Enter the root of the tree\n");
	scanf("%c", &key);
	a = create_root(a, key);
	menu();
	while (ch) {
		printf("-> ");
		scanf("%d", &ch);
		switch (ch) {
		case 0:
			break;
		case 1:
			printf("Enter the node of the tree: \n");
			printf("-> ");
			getchar();
			scanf("%c", &key);
			add_node(a, key);
			break;
		case 2:
			if (a)
				print_tree(a);
			else
				printf("Tree is empty! \n");
			break;
		case 3:
			printf("Enter the node of the tree: \n");
			printf("-> ");
		    getchar();
			scanf("%c", &key);
			remove_node(&a, key);
			break;
		case 4:
			k = 0;
			x = task(a, &k);
			printf("%d\n", x);
		case 5:
			menu();
			break;
		default:
			exit;
			break;
		}
	}
	return 0;
}
```

```
Enter the root of the tree
6
============================
||          What you want to do ?       ||
||           1-add node                 ||
||           2-print tree               ||
||           3-remove node              ||
||           4-curry to task            ||
||           5-menu                     ||
||           0-end                      ||
============================
-> 2
   -6
-> 4
0
============================
||          What you want to do ?       ||
||           1-add node                 ||
||           2-print tree               ||
||           3-remove node              ||
||           4-curry to task            ||
||           5-menu                     ||
||           0-end                      ||
============================
-> 1
Enter the node of the tree:
-> a
-> 1
Enter the node of the tree:
-> b
-> 1
Enter the node of the tree:
-> c
-> 1
Enter the node of the tree:
-> d
```

-> 2

-d

-c

-b

-a

-6

-> 4

4

==========

|| What you want to do ? ||

|| 1-add node ||

|| 2-print tree ||

|| 3-remove node ||

|| 4-curry to task ||

|| 5-menu ||

|| 0-end ||

==========

-> 3

Enter the node of the tree:

-> 6

-> 2

-d

-c

-b

-a

-> 3

Enter the node of the tree:

-> a

-> 3

Enter the node of the tree:

-> b

-> 3

Enter the node of the tree:

-> c

-> 2

-d

-> 1

Enter the node of the tree:

-> b

-> 1

Enter the node of the tree:

-> f

-> 5

==========

|| What you want to do ? ||

|| 1-add node ||

|| 2-print tree ||

|| 3-remove node ||

|| 4-curry to task ||

|| 5-menu ||

|| 0-end ||

==========

-> 2

-f

-d

-b

-> 4

1

==========

|| What you want to do ? ||

|| 1-add node ||

|| 2-print tree ||

|| 3-remove node ||

|| 4-curry to task ||

|| 5-menu ||

|| 0-end ||

==========

-> 0