ASSIGNMENT-10

1. Write a menu driven program to implement Heap Sort using Binary Tree

SOLUTION:

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

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->left = newNode->right = NULL;
    return newNode;
}

struct Node* insertNode(struct Node* root, int data) {
    if (root == NULL)
        return createNode(data);
    if (data <= root->data)
        root->left = insertNode(root->left, data);
    else
        root->right = insertNode(root->right, data);
    return root;
}

void printTree(struct Node* root, int space) {
    if (root == NULL)
        return;
    space += 5;
    printTree(root->right, space);
    printf("\n");
    for (int i = 5; i < space; i++)
        printf(" ");
    printf("%d\n", root->data);
    printTree(root->left, space);
}

struct Node* buildTree(int arr[], int n) {
```

```
    struct Node* root = NULL;
    for (int i = 0; i < n; i++) {
        root = insertNode(root, arr[i]);
    }
    return root;
}

void inOrderTraversal(struct Node* root, int arr[], int* index) {
    if (root != NULL) {
        inOrderTraversal(root->left, arr, index);
        arr[*index] = root->data;
        (*index)++;
        inOrderTraversal(root->right, arr, index);
    }
}

void heapSort(struct Node* root, int arr[], int n) {
    int index = 0;
    inOrderTraversal(root, arr, &index);
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the elements: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    struct Node* root = buildTree(arr, n);
    int choice;
    do {
        printf("\nMENU:\n");
        printf("1. Display Original Binary Tree\n");
        printf("2. Heap Sort (Min Heap - Ascending Order)\n");
        printf("3. Heap Sort (Max Heap - Descending Order)\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
```

```
            printf("\nOriginal Binary Tree: \n");
            printTree(root, 0);
            break;
        case 2:
            heapSort(root, arr, n);
            printf("\nSorted array (Min Heap - Ascending Order): ");
            printArray(arr, n);
            break;
        case 3:
            heapSort(root, arr, n);
            printf("\nSorted array (Max Heap - Descending Order): ");
            for (int i = n - 1; i >= 0; i--)
                printf("%d ", arr[i]);
            printf("\n");
            break;
        case 4:
            printf("\nExiting...\n");
            break;
        default:
            printf("\nInvalid choice. Please choose a valid option.\n");
      }
   } while (choice != 4);
   return 0;
}
```

OUTPUT:

```
Enter the number of elements: 5
Enter the elements: 10 50 30 90 70

MENU:
1. Display Original Binary Tree
2. Heap Sort (Min Heap - Ascending Order)
3. Heap Sort (Max Heap - Descending Order)
4. Exit
Enter your choice: 1

Original Binary Tree:

        90

            70

    50

        30

10
```

MENU:
1. Display Original Binary Tree
2. Heap Sort (Min Heap - Ascending Order)
3. Heap Sort (Max Heap - Descending Order)
4. Exit
Enter your choice: 2

Sorted array (Min Heap - Ascending Order): 10 30 50 70 90

MENU:
1. Display Original Binary Tree
2. Heap Sort (Min Heap - Ascending Order)
3. Heap Sort (Max Heap - Descending Order)
4. Exit
Enter your choice: 3

Sorted array (Max Heap - Descending Order): 90 70 50 30 10