**DATASTRUCTURE**

**PROGRAMS:**

**1.Binary Heap**

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <limits.h>*

*struct MinHeap {*

*int \*harr; // pointer to array of elements in heap*

*int capacity; // maximum possible size of min heap*

*int heap\_size; // current number of elements in min heap*

*};*

*struct MinHeap\* createMinHeap(int capacity);*

*void insertKey(struct MinHeap\* minHeap, int k);*

*void decreaseKey(struct MinHeap\* minHeap, int i, int new\_val);*

*int extractMin(struct MinHeap\* minHeap);*

*void deleteKey(struct MinHeap\* minHeap, int i);*

*void minHeapify(struct MinHeap\* minHeap, int i);*

*int getMin(struct MinHeap\* minHeap);*

*struct MinHeap\* createMinHeap(int capacity) {*

*struct MinHeap\* minHeap = (struct MinHeap\*)malloc(sizeof(struct MinHeap));*

*minHeap->capacity = capacity;*

*minHeap->heap\_size = 0;*

*minHeap->harr = (int\*)malloc(capacity \* sizeof(int));*

*return minHeap;*

*}*

*void insertKey(struct MinHeap\* minHeap, int k) {*

*if (minHeap->heap\_size == minHeap->capacity) {*

*printf("\nOverflow: Could not insertKey\n");*

*return;*

*}*

*int i = minHeap->heap\_size++;*

*minHeap->harr[i] = k;*

*while (i != 0 && minHeap->harr[(i - 1) / 2] > minHeap->harr[i]) {*

*int temp = minHeap->harr[i];*

*minHeap->harr[i] = minHeap->harr[(i - 1) / 2];*

*minHeap->harr[(i - 1) / 2] = temp;*

*i = (i - 1) / 2;*

*}*

*}*

*void decreaseKey(struct MinHeap\* minHeap, int i, int new\_val) {*

*minHeap->harr[i] = new\_val;*

*while (i != 0 && minHeap->harr[(i - 1) / 2] > minHeap->harr[i]) {*

*int temp = minHeap->harr[i];*

*minHeap->harr[i] = minHeap->harr[(i - 1) / 2];*

*minHeap->harr[(i - 1) / 2] = temp;*

*i = (i - 1) / 2;*

*}*

*}*

*int extractMin(struct MinHeap\* minHeap) {*

*if (minHeap->heap\_size <= 0)*

*return INT\_MAX;*

*if (minHeap->heap\_size == 1) {*

*minHeap->heap\_size--;*

*return minHeap->harr[0];*

*}*

*int root = minHeap->harr[0];*

*minHeap->harr[0] = minHeap->harr[minHeap->heap\_size - 1];*

*minHeap->heap\_size--;*

*minHeapify(minHeap, 0);*

*return root;*

*}*

*void deleteKey(struct MinHeap\* minHeap, int i) {*

*decreaseKey(minHeap, i, INT\_MIN);*

*extractMin(minHeap);*

*}*

*void minHeapify(struct MinHeap\* minHeap, int i) {*

*int l = 2 \* i + 1;*

*int r = 2 \* i + 2;*

*int smallest = i;*

*if (l < minHeap->heap\_size && minHeap->harr[l] < minHeap->harr[smallest])*

*smallest = l;*

*if (r < minHeap->heap\_size && minHeap->harr[r] < minHeap->harr[smallest])*

*smallest = r;*

*if (smallest != i) {*

*int temp = minHeap->harr[i];*

*minHeap->harr[i] = minHeap->harr[smallest];*

*minHeap->harr[smallest] = temp;*

*minHeapify(minHeap, smallest);*

*}*

*}*

*int getMin(struct MinHeap\* minHeap) {*

*return minHeap->harr[0];*

*}*

*int main() {*

*struct MinHeap\* minHeap = createMinHeap(11);*

*insertKey(minHeap, 3);*

*insertKey(minHeap, 2);*

*deleteKey(minHeap, 1);*

*insertKey(minHeap, 15);*

*insertKey(minHeap, 5);*

*insertKey(minHeap, 4);*

*insertKey(minHeap, 45);*

*printf("%d ", extractMin(minHeap));*

*printf("%d ", getMin(minHeap));*

*decreaseKey(minHeap, 2, 1);*

*printf("%d", getMin(minHeap));*

*return 0;*

*}*

**OUTPUT:**

*2 4 1*

**2.Binary Sort**

*#include <stdio.h>*

*void printArray(int arr[], int n) {*

*for (int i = 0; i < n; i++)*

*printf("%d ", arr[i]);*

*printf("\n");*

*}*

*int binarySearch(int arr[], int item, int low, int high) {*

*if (high <= low)*

*return (item > arr[low]) ? (low + 1) : low;*

*int mid = (low + high) / 2;*

*if (item == arr[mid])*

*return mid + 1;*

*if (item > arr[mid])*

*return binarySearch(arr, item, mid + 1, high);*

*return binarySearch(arr, item, low, mid - 1);*

*}*

*void binaryInsertionSort(int arr[], int n) {*

*int i, j, selected, loc;*

*for (i = 1; i < n; ++i) {*

*j = i - 1;*

*selected = arr[i];*

*loc = binarySearch(arr, selected, 0, j);*

*while (j >= loc) {*

*arr[j + 1] = arr[j];*

*j--;*

*}*

*arr[j + 1] = selected;*

*}*

*}*

*int main() {*

*int arr[] = {37, 23, 0, 17, 12, 72, 31, 46, 100, 88, 54};*

*int n = sizeof(arr) / sizeof(arr[0]);*

*printf("Original array: \n");*

*printArray(arr, n);*

*binaryInsertionSort(arr, n);*

*printf("\nSorted array: \n");*

*printArray(arr, n);*

*return 0;*

*}*

**OUTPUT:**

*Original array:*

*37 23 0 17 12 72 31 46 100 88 54*

*Sorted array:*

*0 12 17 23 31 37 46 54 72 88 100*