Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
(Autonomous Institute Affiliated to University of Mumbai)

Name: Jhaveri Varun Nimitt

UID: 2023800042

Batch: CSE A Batch C

Experiment No.:9

Aim: B Heap

Problem:

Create Heap using Successive insertion operation

Increase/ Decrease key operation

Heapsort

Solution:

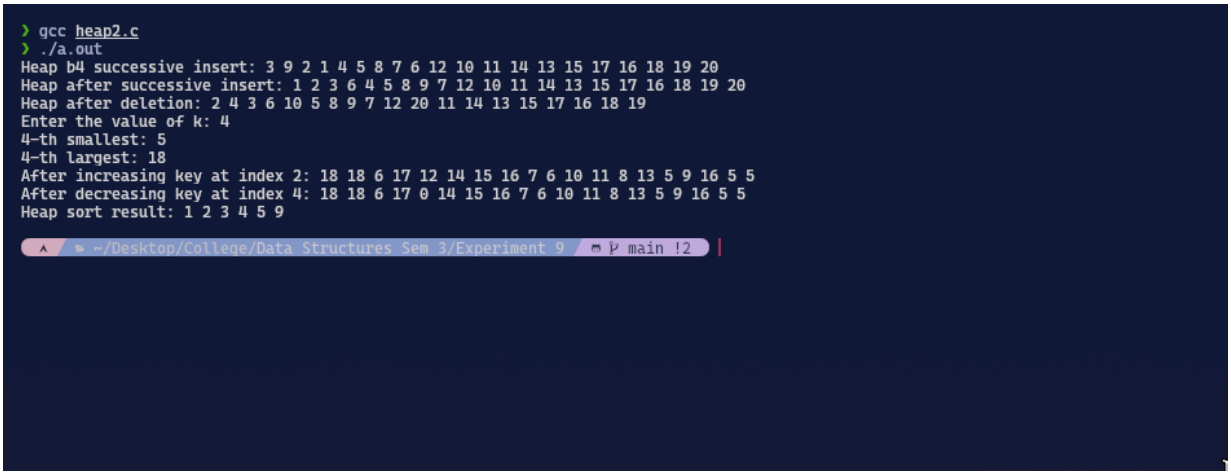
```c
#include <stdio.h>
#include <stdlib.h>

void swap(int *a, int *b);
void minheap(int arr[], int n, int i);
void maxheap(int arr[], int n, int i);
void bobthebuildersuccesiveinsert(int arr[], int n);
void increaseKey(int arr[], int i, int new_val);
void decreaseKey(int arr[], int n, int i, int new_val);
int deleteRoot(int arr[], int *n, int isMinHeap);
int smallest(int arr[], int n, int k);
int largest(int arr[], int n, int k);
void sorts(int arr[], int n);
void printArray(int arr[], int n);

int smallest(int arr[], int n, int k) {
    bobthebuildersuccesiveinsert(arr, n);
    for (int i = 0; i < k - 1; i++)
        deleteRoot(arr, &n, 1);
    return arr[0];
}

int largest(int arr[], int n, int k) {
    for (int i = n / 2 - 1; i >= 0; i--)
        maxheap(arr, n, i);
    for (int i = 0; i < k - 1; i++)
        deleteRoot(arr, &n, 0);
    return arr[0];
}

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void minheap(int arr[], int n, int i) {
    int smallest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] < arr[smallest])
        smallest = left;
    if (right < n && arr[right] < arr[smallest])
        smallest = right;
```

```c
    if (smallest != i) {
        swap(&arr[i], &arr[smallest]);
        minheap(arr, n, smallest);
    }
}

void maxheap(int arr[], int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < n && arr[left] > arr[largest])
        largest = left;
    if (right < n && arr[right] > arr[largest])
        largest = right;
    if (largest != i) {
        swap(&arr[i], &arr[largest]);
        maxheap(arr, n, largest);
    }
}

int deleteRoot(int arr[], int *n, int isMinHeap) {
    if (*n <= 0) return -1;
    int root = arr[0];
    arr[0] = arr[*n - 1];
    (*n)--;
    if (isMinHeap)
        minheap(arr, *n, 0);
    else
        maxheap(arr, *n, 0);
    return root;
}

void sorts(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--)
        maxheap(arr, n, i);
    for (int i = n - 1; i > 0; i--) {
        swap(&arr[0], &arr[i]);
        maxheap(arr, i, 0);
    }
}

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

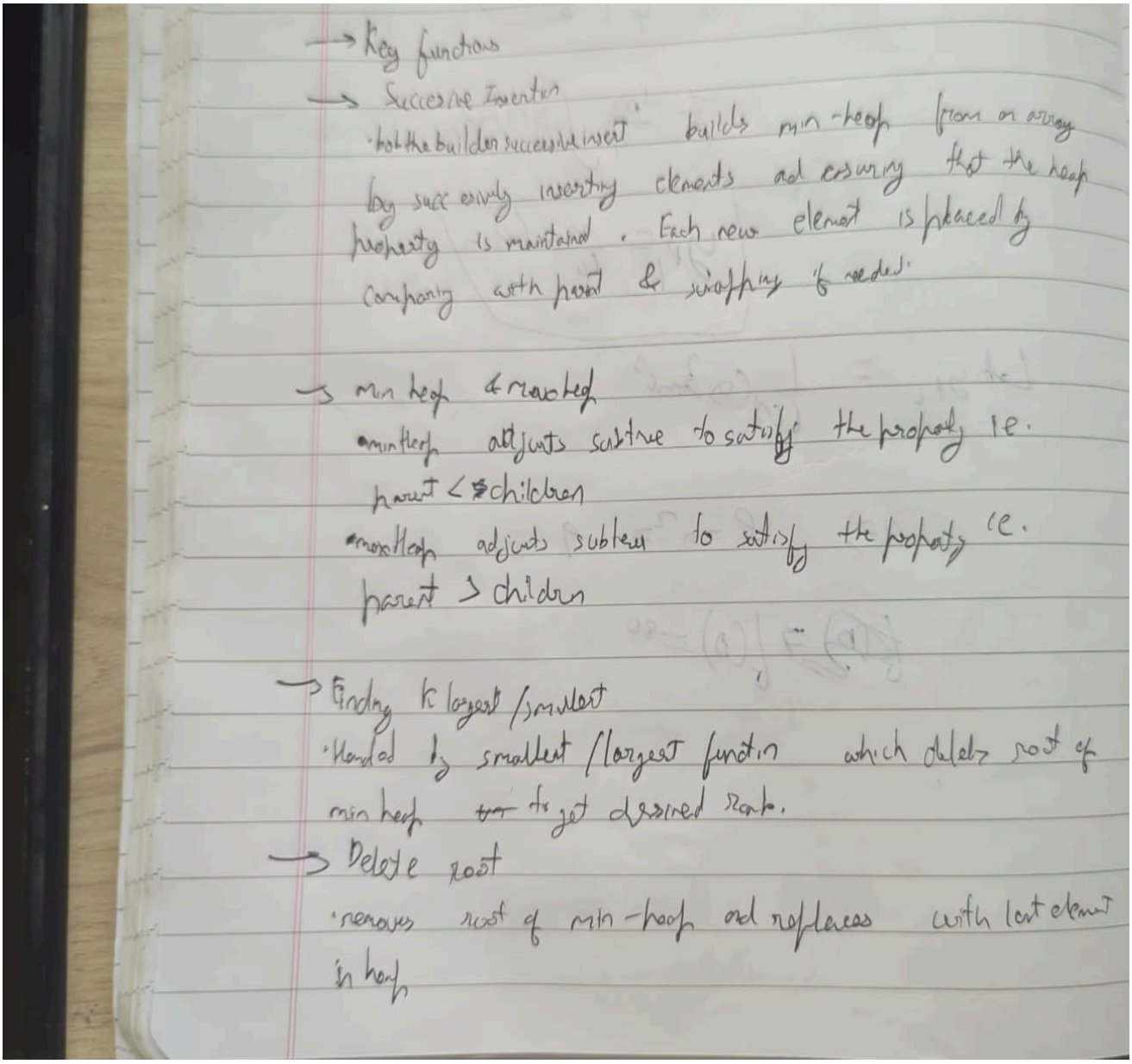
```c
void bobthebuildersuccesiveinsert(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int j = i;
        while (j > 0 && arr[j] < arr[(j - 1) / 2]) {
            swap(&arr[j], &arr[(j - 1) / 2]);
            j = (j - 1) / 2;
        }
    }
}

void increaseKey(int arr[], int i, int new_val) {
    arr[i] = new_val;
    while (i > 0 && arr[i] > arr[(i - 1) / 2]) {
        swap(&arr[i], &arr[(i - 1) / 2]);
        i = (i - 1) / 2;
    }
}

void decreaseKey(int arr[], int n, int i, int new_val) {
    arr[i] = new_val;
    minheap(arr, n, i);
}

int main() {
    int arr[] = {3, 9, 2, 1, 4, 5, 8, 7, 6, 12, 10, 11, 14, 13, 15, 17, 16, 18, 19, 20};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Heap b4 successive insert: ");
    printArray(arr, n);
    bobthebuildersuccesiveinsert(arr, n);
    printf("Heap after successive insert: ");
    printArray(arr, n);

    deleteRoot(arr, &n, 1);
    printf("Heap after deletion: ");
    printArray(arr, n);

    int k;
    printf("Enter the value of k: ");
    scanf("%d", &k);

    printf("%d-th smallest: %d\n", k, smallest(arr, n, k));
    printf("%d-th largest: %d\n", k, largest(arr, n, k));

    increaseKey(arr, 2, 6);
    printf("After increasing key at index 2: ");
    printArray(arr, n);
```

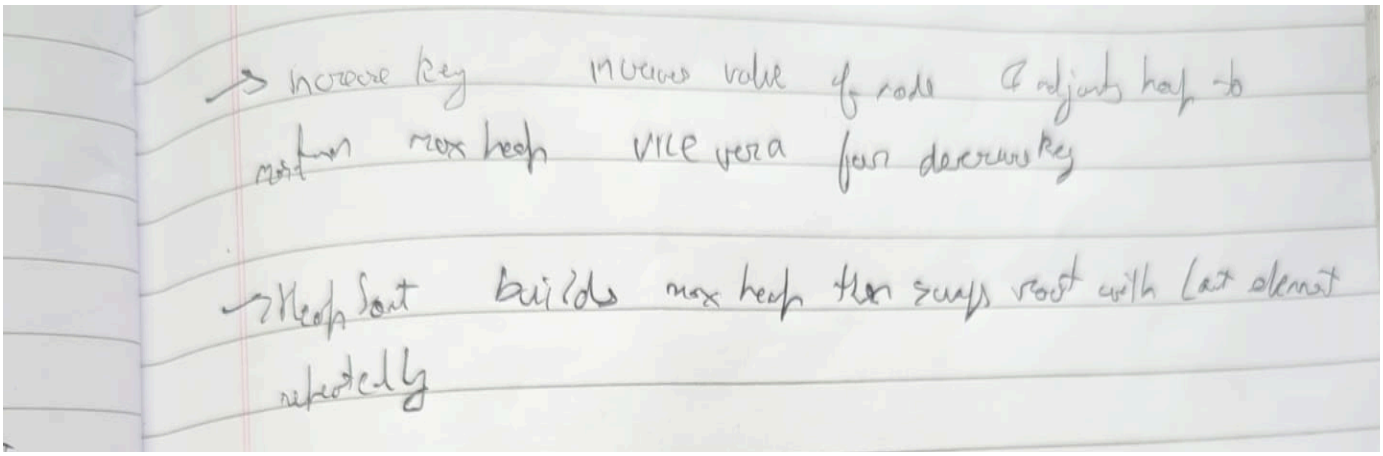
```
    decreaseKey(arr, n, 4, 0);
    printf("After decreasing key at index 4: ");
    printArray(arr, n);

    int arr2[] = {3, 9, 2, 1, 4, 5};
    int m = sizeof(arr2) / sizeof(arr2[0]);
    sorts(arr2, m);
    printf("Heap sort result: ");
    printArray(arr2, m);

    return 0;
}
```

OUTPUT :


❯ gcc heap2.c
❯ ./a.out
Heap b4 successive insert: 3 9 2 1 4 5 8 7 6 12 10 11 14 13 15 17 16 18 19 20
Heap after successive insert: 1 2 3 6 4 5 8 9 7 12 10 11 14 13 15 17 16 18 19 20
Heap after deletion: 2 4 3 6 10 5 8 9 7 12 20 11 14 13 15 17 16 18 19
Enter the value of k: 4
4-th smallest: 5
4-th largest: 18
After increasing key at index 2: 18 18 6 17 12 14 15 16 7 6 10 11 8 13 5 9 16 5 5
After decreasing key at index 4: 18 18 6 17 0 14 15 16 7 6 10 11 8 13 5 9 16 5 5
Heap sort result: 1 2 3 4 5 9
~/Desktop/College/Data Structures Sem 3/Experiment 9
main !2

Handwritten stuff:




→ increase key moves value of node & adjusts heap to
maintain max heap vice versa for decrease key
→ Heap Sort builds max heap then swaps root with last element
recursively