**K23-0607 DS LAB # 10 Nov 14,2024**

**Question # 1**

#include <iostream>

using namespace std;

void swap(int &a, int &b){

    int temp = a;

    a = b;

    b = temp;

}

void min\_heapify(int \*arr, int n, int i) {

    int index = i;

    int left = 2 \* i + 1;

    int right = 2 \* i + 2;

    if (left < n && arr[left] < arr[index]) {

        index = left;

    }

    if (right < n && arr[right] < arr[index]) {

        index = right;

    }

    if (index != i) {

        swap(arr[i], arr[index]);

        min\_heapify(arr, n, index);

    }

}

void Build\_Min\_heap(int \*arr, int n) {

    for (int i = n / 2 - 1; i >= 0; i--) {

        min\_heapify(arr, n, i);

    }

}

void max\_heapify(int \*arr, int n, int i) {

    int index = i;

    int left = 2 \* i + 1;

    int right = 2 \* i + 2;

    if (left < n && arr[left] > arr[index]) {

        index = left;

    }

    if (right < n && arr[right] > arr[index]) {

        index = right;

    }

    if (index != i) {

        swap(arr[i], arr[index]);

        max\_heapify(arr, n, index);

    }

}

void Build\_Max\_Heap(int \*arr, int n) {

    for (int i = n / 2 - 1; i >= 0; i--) {

        max\_heapify(arr, n, i);

    }

}

int main() {

    int arr1[] = {7, 1, 6, 2, 5, 9, 10, 2};

    int n1 = sizeof(arr1) / sizeof(arr1[0]);

    Build\_Min\_heap(arr1, n1);

    cout << "Min Heap : ";

    for (int i = 0; i < n1; i++) {

        cout << arr1[i] << " ";

    }

    cout << endl;

    int arr2[] = {7, 1, 6, 2, 5, 9, 10, 2};

    int n2 = sizeof(arr2) / sizeof(arr2[0]);

    Build\_Max\_Heap(arr2, n2);

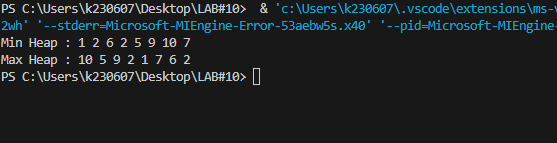
    cout << "Max Heap : ";

    for (int i = 0; i < n2; i++) {

        cout << arr2[i] << " ";

    }

}

****

**Question # 2**

#include <iostream>

using namespace std;

void swap(int &a, int &b){

    int temp = a;

    a = b;

    b = temp;

}

void min\_heapify(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;

    }

    if (smallest != i) {

        swap(arr[i], arr[smallest]);

        min\_heapify(arr, n, smallest);

    }

}

void Build\_Min\_Heap(int arr[], int n) {

    for (int i = n / 2 - 1; i >= 0; i--) {

        min\_heapify(arr, n, i);

    }

}

void max\_heapify(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]);

        max\_heapify(arr, n, largest);

    }

}

void Build\_Max\_Heap(int arr[], int n) {

    for (int i = n / 2 - 1; i >= 0; i--) {

        max\_heapify(arr, n, i);

    }

}

void Delete\_Root(int arr[], int& n) {

    arr[0] = arr[n - 1];

    n--;

}

void HeapSort(int arr[], int n) {

    Build\_Max\_Heap(arr, n);

    for (int i = n - 1; i > 0; i--) {

        swap(arr[0], arr[i]);

        n--;

        max\_heapify(arr, n, 0);

    }

}

void print(int \*arr, int n){

    for (int i = 0; i < n; i++) {

        cout << arr[i] << " ";

    }

    cout << endl;

}

int main() {

    int arr[] = {35, 33, 42, 10, 14, 19, 27, 44, 26, 31};

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

    Build\_Min\_Heap(arr, n);

    cout << "Build Min Heap : ";

    print(arr,n);

    Delete\_Root(arr, n);

    Build\_Max\_Heap(arr, n);

    cout << "Build Max Heap : : " ;

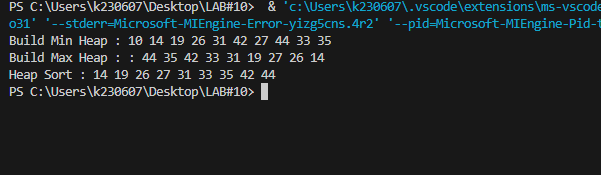
    print(arr,n);

    HeapSort(arr, n);

    cout << "Heap Sort : ";

    print(arr,n);

}

****

**Question # 3**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void swap(int &a, int &b){

    int temp = a;

    a = b;

    b = temp;

}

class task {

    public:

    int id;

    int priority;

};

void max\_heapify(task arr[], int n, int i) {

    int largest = i;

    int left = 2 \* i + 1;

    int right = 2 \* i + 2;

    if (left < n && arr[left].priority > arr[largest].priority) {

        largest = left;

    }

    if (right < n && arr[right].priority > arr[largest].priority) {

        largest = right;

    }

    if (largest != i) {

        swap(arr[i], arr[largest]);

        max\_heapify(arr, n, largest);

    }

}

void Build\_Max\_Heap(task arr[], int n) {

    for (int i = n / 2 - 1; i >= 0; i--) {

        max\_heapify(arr, n, i);

    }

}

void Delete\_Root(task arr[], int& n) {

    arr[0] = arr[n - 1];

    n--;

    max\_heapify(arr, n, 0);

}

int main() {

    srand(time(0));

    int n = 10;

    task arr[10];

    for (int i = 0; i < n; i++) {

        arr[i].id = i + 1;

        arr[i].priority = rand() % 10 + 1;

    }

    Build\_Max\_Heap(arr, n);

    cout << "Tasks (by priority):" << endl;

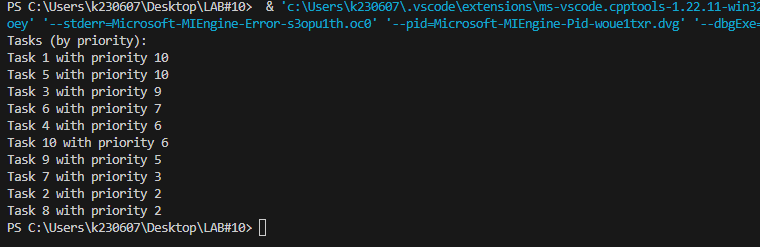
    while (n > 0) {

        cout << "Task " << arr[0].id << " with priority " << arr[0].priority << endl;

        Delete\_Root(arr, n);

    }

}

****

**END**