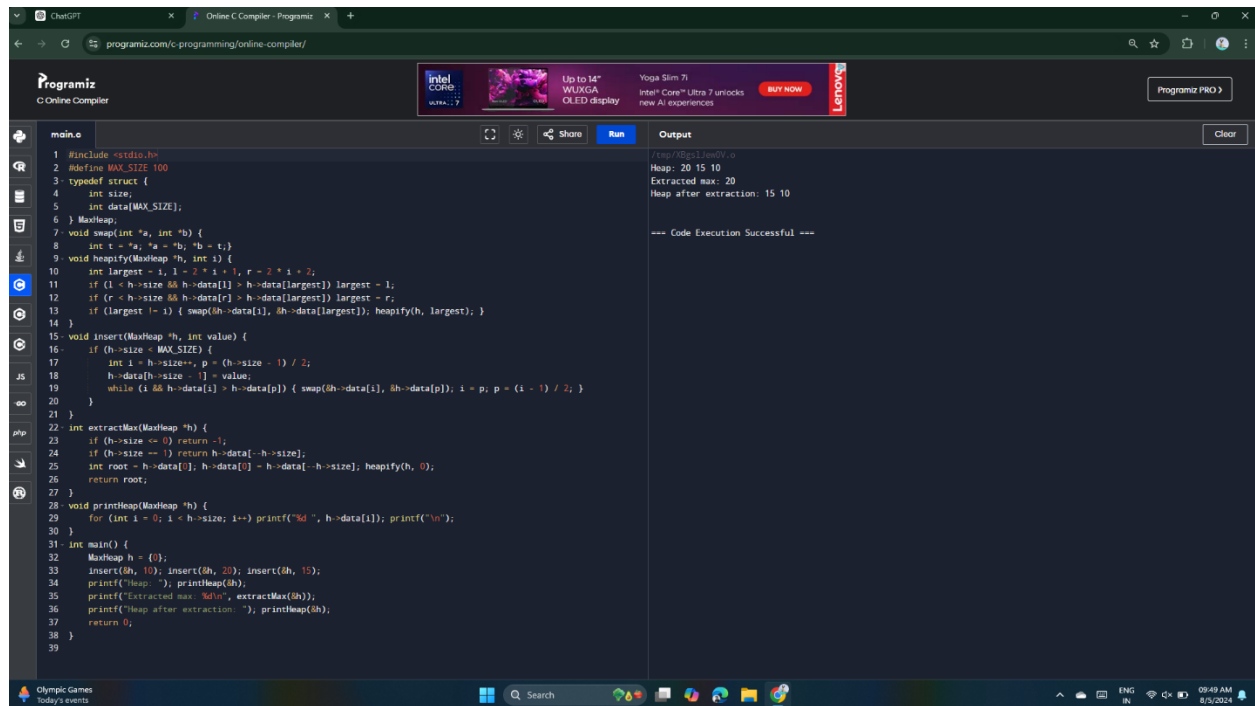


1.C code for binary heap



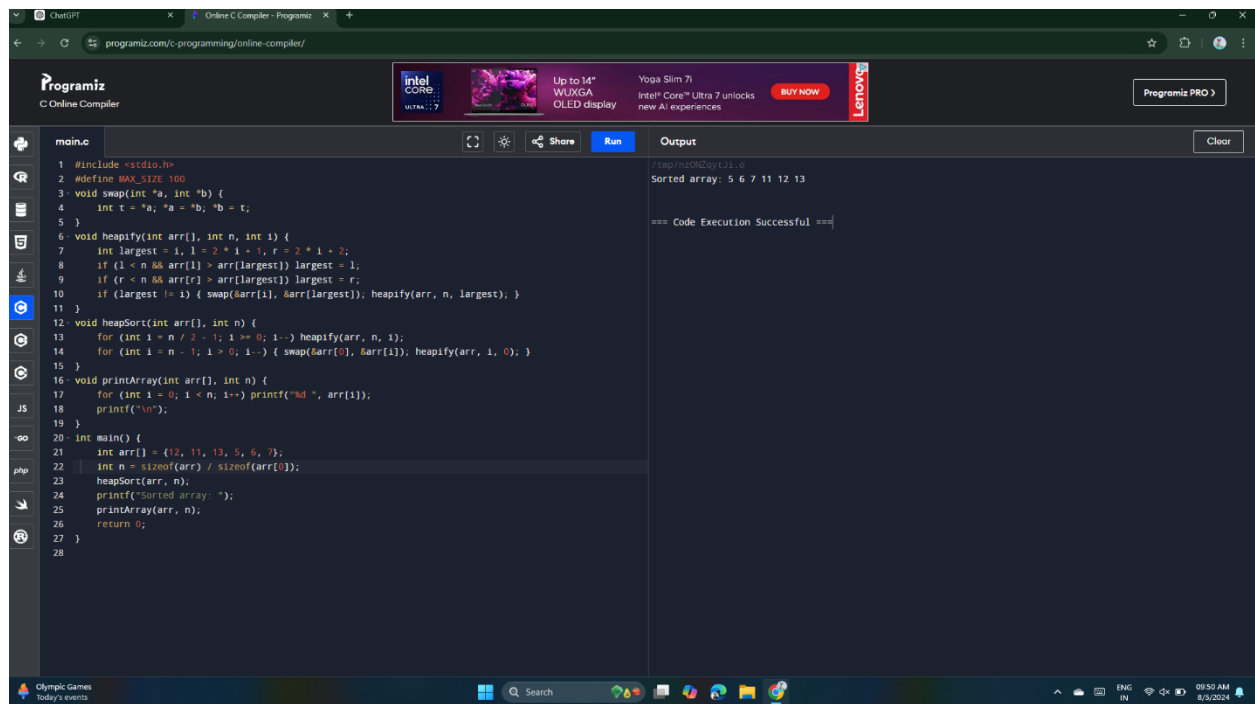
```
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 typedef struct {
4     int size;
5     int data[MAX_SIZE];
6 } MaxHeap;
7 void swap(int *a, int *b) {
8     int t = *a; *a = *b; *b = t;
9 }
10 void heapify(MaxHeap *h, int i) {
11     int largest = i, l = 2 * i + 1, r = 2 * i + 2;
12     if (l < h->size && h->data[l] > h->data[largest]) largest = l;
13     if (r < h->size && h->data[r] > h->data[largest]) largest = r;
14     if (largest != i) { swap(&h->data[i], &h->data[largest]); heapify(h, largest); }
15 }
16 void insert(MaxHeap *h, int value) {
17     if (h->size == MAX_SIZE) {
18         return;
19     }
20     int i = h->size++;
21     h->data[i] = value;
22     while (i > 0 && h->data[i] > h->data[(i-1)/2]) {
23         swap(&h->data[i], &h->data[(i-1)/2]);
24         i = (i-1)/2;
25     }
26 }
27 int extractMax(MaxHeap *h) {
28     if (h->size == 0) return -1;
29     if (h->size == 1) return h->data[0];
30     int root = h->data[0];
31     h->data[0] = h->data[h->size-1];
32     heapify(h, 0);
33     return root;
34 }
35 void printHeap(MaxHeap *h) {
36     for (int i = 0; i < h->size; i++) printf("%d ", h->data[i]); printf("\n");
37 }
38 int main() {
39     MaxHeap h = {0};
40     insert(&h, 10); insert(&h, 20); insert(&h, 15);
41     printf("Heap: "); printHeap(&h);
42     printf("Extracted max: %d\n", extractMax(&h));
43     printf("Heap after extraction: "); printHeap(&h);
44     return 0;
45 }
```

Output:

```
//tmp/8Bp1Jw0V.o
Heap: 20 15 10
Extracted max: 20
Heap after extraction: 15 10

=== Code Execution Successful ===
```

2.Heap Sorting in Ascending Order



```
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 void swap(int *a, int *b) {
4     int t = *a; *a = *b; *b = t;
5 }
6 void heapify(int arr[], int n, int i) {
7     int largest = i, l = 2 * i + 1, r = 2 * i + 2;
8     if (l < n && arr[l] > arr[largest]) largest = l;
9     if (r < n && arr[r] > arr[largest]) largest = r;
10    if (largest != i) { swap(&arr[i], &arr[largest]); heapify(arr, n, largest); }
11 }
12 void heapSort(int arr[], int n) {
13     for (int i = n / 2 - 1; i >= 0; i--) heapify(arr, n, i);
14     for (int i = n - 1; i > 0; i--) { swap(&arr[0], &arr[i]); heapify(arr, i, 0); }
15 }
16 void printArray(int arr[], int n) {
17     for (int i = 0; i < n; i++) printf("%d ", arr[i]);
18     printf("\n");
19 }
20 int main() {
21     int arr[] = {12, 11, 13, 5, 6, 7};
22     int n = sizeof(arr) / sizeof(arr[0]);
23     heapSort(arr, n);
24     printf("Sorted array: ");
25     printArray(arr, n);
26     return 0;
27 }
```

Output:

```
//tmp/m20C2qjCj1.o
Sorted array: 5 6 7 11 12 13

=== Code Execution Successful ===
```



Edit with WPS Office

3. Heap Sorting in Descending Order

```
C Online Compiler
17,799.00 16,099.00
main.c
Run
Output
Clear

1 #include <stdio.h>
2 void heapify(int arr[], int n, int i) {
3     int largest = i; // Initialize largest as root
4     int left = 2 * i + 1; // Left child
5     int right = 2 * i + 2; // Right child
6     if (left < n && arr[left] > arr[largest])
7         largest = left;
8     if (right < n && arr[right] > arr[largest])
9         largest = right;
10    if (largest != i) {
11        int temp = arr[i];
12        arr[i] = arr[largest];
13        arr[largest] = temp;
14        heapify(arr, n, largest);
15    }
16    void heapSort(int arr[], int n) {
17        for (int i = n / 2 - 1; i >= 0; i--)
18            heapify(arr, n, i);
19        for (int i = n - 1; i > 0; i--) {
20            int temp = arr[0];
21            arr[0] = arr[i];
22            arr[i] = temp;
23            heapify(arr, i, 0);
24        }
25    }
26    void printArray(int arr[], int n) {
27        for (int i = 0; i < n; i++)
28            printf("%d ", arr[i]);
29        printf("\n");
30    }
31    int main() {
32        int arr[] = {12, 11, 13, 5, 6, 7};
33        int n = sizeof(arr) / sizeof(arr[0]);
34        heapSort(arr, n);
35        printf("Sorted array in descending order is: ");
36        printArray(arr, n);
37        return 0;
38    }
39
40
```

Sorted array in descending order is: 5 6 7 11 12 13

== Code Execution Successful ==



Edit with WPS Office