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

#define MAX_SIZE 15

// returns the index of the parent node
int parent(int i) {
    return (i - 1) / 2;
}

// return the index of the left child
int left_child(int i) {
    return 2*i + 1;
}

// return the index of the right child
int right_child(int i) {
    return 2*i + 2;
}

void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}

// insert the item at the appropriate position
void insert(int a[], int data, int *n) {
    if (*n >= MAX_SIZE) {
        printf("%s\n", "The heap is full. Cannot insert");
        return;
    }
    // first insert the time at the last position of the array
    // and move it up
    a[*n] = data;
    *n = *n + 1;


    // move up until the heap property satisfies
    int i = *n - 1;
    while (i != 0 && a[parent(i)] < a[i]) {
        swap(&a[parent(i)], &a[i]);
        i = parent(i);
    }
}

// moves the item at position i of array a
// into its appropriate position
void max_heapify(int a[], int i, int n){
    // find left child node
    int left = left_child(i);

    // find right child node
    int right = right_child(i);

    // find the largest among 3 nodes
    int largest = i;

    // check if the left node is larger than the current node
    if (left <= n && a[left] > a[largest]) {
        largest = left;
    }

    // check if the right node is larger than the current node
    if (right <= n && a[right] > a[largest]) {
        largest = right;
    }

    // swap the largest node with the current node
    // and repeat this process until the current node is larger
    // the right and the left node
    if (largest != i) {
        int temp = a[i];
        a[i] = a[largest];
        a[largest] = temp;
        max_heapify(a, largest, n);
    }

```

```
}

// converts an array into a heap
void build_max_heap(int a[], int n) {
    int i;
    for (i = n/2; i >= 0; i--) {
        max_heapify(a, i, n);
    }
}

// returns the  maximum item of the heap
int get_max(int a[]) {
    return a[0];
}

// deletes the max item and return
int extract_max(int a[], int *n) {
    int max_item = a[0];

    // replace the first item with the last item
    a[0] = a[*n - 1];
    *n = *n - 1;

    // maintain the heap property by heapifying the
    // first item
    max_heapify(a, 0, *n);
    return max_item;
}

// prints the heap
void print_heap(int a[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        printf("%d\n", a[i]);
    }
    printf("\n");
}
```

```
int main() {
    int n = 10;
    int a[MAX_SIZE];
    a[1] = 10; a[2] = 12; a[3] = 9; a[4] = 78; a[5] = 33; a[6] = 2
    build_max_heap(a, n);
    insert(a, 55, &n);
    insert(a, 56, &n);
    insert(a, 57, &n);
    insert(a, 58, &n);
    insert(a, 100, &n);
    print_heap(a, n);
    return 0;
}
```