Shubham Singh /2022300118/ Batch C / Comps B

Exp-9: implement max heap and its different operations

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

Maxheap* initheap (int capacity) {
    Maxheap* newHeap = malloc (sizeof (Maxheap));
    newheap -> capacity = capacity;
    newheap -> size = 0;
    newheap -> array = calloc (capacity+1, sizeof(int));
    return newheap;
}

void destroyheap (MaxHeap* heap) {
    free (heap -> array);
    free (heap);
}

void Heapify (Maxheap* heap, int i) {
    int * parent, * right, * left
    parent = heap -> array + i;
    if (2*i <= heap -> size) {
        left = heap -> array + 2*i;
    } else { left = NULL; }
    if ((2*i+1) <= heap -> size) {
        right = heap -> array + (2*i+1);
    } else { right = NULL; }
```

→

```
if (left == NULL && right == NULL) {
 return ;}
if (left == NULL) {
    if (*right > *parent) {
        swap(parent ,right);
        heapify(heap, 2*i+1);
    } else { return; }
else if (right == NULL) {
    if (*left > *parent) {
        swap (parent, left);
        heapify (heap, 2*i);
    } return {return;}
else {
    int* max;
    if (*parent > *left) { max = parent;} }
    else { max = left; }
    if (*parent < *right) { max = right; }
    if (max == parent) { return;}
    if (max == right) {
        swap (right, parent);
        heapify (heap, 2*i+1);
    }
    else {
        swap (left, parent);
        heapify (heap, 2*i);
    }
}
}
```

```
void insert (MaxHeap* heap, int value) {
    if (heap->size == heap->capacity) {
        printf ("Heap is full\n");
        return;
    }
    heap->size++;
    heap->array[heap->size] = value;
    int parent = heap->size/2;
    while (parent >= 1) {
        heapify (heap, parent);
        parent = parent/2;
    }
}

void peek_max (MaxHeap* heap) {
    if (heap->size == 0) {
        printf ("Heap is empty!");
        return;
    }
    printf ("The maximum element in the given maxheap
            array is: %d\n", heap->array[0]);
}

void extractMax (MaxHeap* heap) {
    int retval = heap->array[1];
    heap->array[1] = heap->array[heap->size];
    heap->size--;
    if (heap->size > 1) {
        heapify (heap, 1);
    }
    return retval;
}
```

```
void display_heap (MaxHeap * heap, int stop_idx) {
    printf ("[");
    for (int i=1 ; i < stop_idx ; i++) {
        printf ("%d, ", heap -> array [i] ;
    }
    printf ("%d]\n", heap -> array [stop_idx]);
}

MaxHeap* Construct_heap (int * arr, int arr_length) {
    MaxHeap* heap = initheap (arr_length);
    for (int i = 0; i< arr_length ; i++) {
        heap -> array [i+1] = arr [i] ;
    }
    heap -> size = arr_length ;
    for (int i= arr_length /2 ; i >= 1 ; i--) {
        heapify (heap, i);
    }

    return heap ;
}
void Heapsort_ascending (MaxHeap* Heap) {
    int initialsize = heap -> size
    swap (& (heap -> array [1] , & heap -> array [heap -> size])

    heap -> size -- ;
    for (int i=0 ; i < initialsize -1 ; i++) {
        heap (heap, 1);
        swap (& heap -> array [1], & (heap -> array [heap -> size]));
        heap -> size -- ;
    }
    heap -> size = initialsize ;
}
```