Structures

● Graded

Student

SRUTHI SUBRAMANIAN

Total Points

200 / 200 pts

Autograder Score

200.0 / 200.0

Passed Tests

Test 1 (10/10)
Test 10 (20/20)
Test 11 (20/20)
Test 12 (20/20)
Test 13 (20/20)
Test 14 (20/20)
Test 2 (10/10)
Test 3 (10/10)
Test 4 (10/10)
Test 5 (10/10)
Test 6 (10/10)
Test 7 (10/10)
Test 8 (10/10)
Test 9 (20/20)

Autograder Results

Test 1 (10/10)

Test 10 (20/20)

Test 11 (20/20)

Test 12 (20/20)

Test 13 (20/20)

Test 14 (20/20)

Test 2 (10/10)

Test 3 (10/10)

Test 4 (10/10)

Test 5 (10/10)

Test 6 (10/10)

Test 7 (10/10)

Test 8 (10/10)

Test 9 (20/20)

Submitted Files

hw.c Download

```c
#include <stdio.h>

int main() {
    // Write C code here
    int o, MAX;
    MAX = 0;
    scanf("%d %d", &o, &MAX);
    if (o == 0) {
        //stack
        int a, b, i;
        int top = 0;
        //top is 1 index above the highest element: when it is MAX, the stack is full
        a = -1;
        while (a != 3) {
            scanf("%d", &a);
            int stack[MAX + 2];
            if (a == 0) {
                //print the stack - bottom to top
                if (top > 0) {
                    for (i = 0; i < top; i++) {
                        printf("%d ", stack[i]);
                    }
                }
                printf("\n");
            }
            if (a == 1) {
                scanf("%d", &b);
                stack[top] = b;
                //push b to the stack
                if (top == MAX) {
                    //stack is full
                    printf("-1\n");
                    break;
                }
                else {
                    top = top + 1;
                }
            }
            if (a == 2) {
                //pop
                if (top == 0) {
                    printf("-1\n");
                    break;
                }
                top = top - 1;
                printf("%d\n", stack[top]);
                stack[top] = 0;
            }
        }
    }
    else if (o == 1) {
        int queue[MAX];
        int a, b, i, head, tail;
        head = 0;
        tail = 0;
        b = 0;
        while (a != 3) {
            scanf("%d ", &a);
            if (a == 0) {
                //print the queue
                if (tail != head) {
                    //the queue is not empty
                    if (head < tail) {
                        for (i = head; i < tail; i++) {
                            printf("%d ", queue[i]);
                        }
                    }
                    else {
                        for (i = head; i < MAX; i++) {
                            printf("%d ", queue[i]);
                        }
                        for (i = 0; i < tail; i++) {
                            printf("%d ", queue[i]);
                        }
                    }
                }
                printf("\n");
            }
            else if (a == 1) {
                //enqueue b
                scanf("%d\n", &b);
                if (head == tail) {
                    queue[head] = b;
                    tail = tail + 1;
                }
                else if (head < tail) {
                    if (head == 0) {
                        if (tail == MAX - 1) {
                            printf("-1\n");
                            break;
                        }
                        else {
                            tail = tail + 1;
                            queue[tail - 1] = b;
                        }
                    }
                    else {
                        if (tail == MAX - 1) {
                            tail = 0;
                            queue[MAX - 1] = b;
                        }
                        else {
                            tail = tail + 1;
                            queue[tail - 1] = b;
                        }
                    }
                }
                else {
                    //head>=tail
                    if (tail + 1 == head) {
                        printf("-1\n");
                        break;
                    }
                    else {
                        tail = tail + 1;
                        queue[tail - 1] = b;
                    }
                }
            }
            if (a == 2) {
                //dequeue - first check if empty
                if (head == tail) {
                    printf("-1\n");
                    break;
                }
                else if (head == MAX - 1) {
                    int temp;
                    temp = queue[head];
                    head = 0;
                    printf("%d\n", temp);
                }
                else {
                    head = head + 1;
                    printf("%d\n", queue[head - 1]);
                }
            }
        }
    }
    else if (o == 2) {
        int a, i;
        int* heap;
        a = -1;
        int size; //the size of the heap array
        size = 0;
        int heapp[MAX];
        heap = heapp;
        int* MinHeapify(int* heap, int ind) {
            //1 indexing!!
            int minind = ind;
            if (ind * 2 <= size && heap[2 * ind - 1] < heap[ind - 1]) {
                minind = 2 * ind;
            }
            if (ind * 2 + 1 <= size && heap[2 * ind] < heap[minind - 1]) {
```

```c
            minind = 2 * ind + 1;
        }
        if (minind != ind) {
            //swap minind term and X and then call MinHeapify on minind
            int X = ind;
            int t = heap[X - 1];
            heap[X - 1] = heap[minind - 1];
            heap[minind - 1] = t;
            return MinHeapify(heap, minind);
        }
        else {
            return heap;
        }
    }
    while (a != 4) {
        scanf("%d ", &a);
        if (a == 0) {
            //print the heap in the usual order of the array
            for (i = 0; i < size; i++) {
                printf("%d ", heap[i]);
            }
            printf("\n");
        }
        else if (a == 1) {
            //build-heap
            int a2;
            int b;
            scanf("%d ", &a2);
            size = a2;
            if (a2 > MAX) {
                printf("-1\n");
                break;
            }
            for (i = 0; i < a2; i++) {
                scanf("%d ", &b);
                heap[i] = b;
            }
            for (i = size; i > 0; i--) {
                heap = MinHeapify(heapp, i);
            }
        }
        else if (a == 2) {
            //decrease-key
            int a2, a3;
            scanf("%d %d", &a2, &a3);
            if (a2 <= size) {
                if (heap[a2 - 1] > a3) {
                    heap[a2 - 1] = a3;
                    for (i = size; i > 0; i--) {
                        heap = MinHeapify(heapp, i);
                    }
                }
                }
            }
            else if (a == 3) {
                //extract minimum
                if (size == 0) {
                    printf("-1\n");
                    break;
                }
                int min = heap[0];
                size = size - 1;
                heap[0] = heap[size];
                heap[size] = min;
                heap = MinHeapify(heapp, 1);
                printf("%d\n", min);
            }
        }
    }


    return 0;
}
```