#include <stdio.h>

#include <stdlib.h>

struct Heap {

int\* arr;

int size;

int capacity;

int isMax;

};

typedef struct Heap heap;

heap\* createHeap(int capacity, int isMax);

void heapify(heap\* h, int index);

void insert(heap\* h, int data);

int extract(heap\* h);

void printHeap(heap\* h);

heap\* createHeap(int capacity, int isMax) {

heap\* h = (heap\*)malloc(sizeof(heap));

h->size = 0;

h->capacity = capacity;

h->arr = (int\*)malloc(capacity \* sizeof(int));

h->isMax = isMax;

return h;

}

void heapify(heap\* h, int index) {

int left = 2 \* index + 1, right = 2 \* index + 2, target = index;

if (left < h->size &&

((h->isMax && h->arr[left] > h->arr[target]) ||

(!h->isMax && h->arr[left] < h->arr[target])))

target = left;

if (right < h->size &&

((h->isMax && h->arr[right] > h->arr[target]) ||

(!h->isMax && h->arr[right] < h->arr[target])))

target = right;

if (target != index) {

int temp = h->arr[target];

h->arr[target] = h->arr[index];

h->arr[index] = temp;

heapify(h, target);

}

}

void insert(heap\* h, int data) {

if (h->size == h->capacity) {

printf("Heap is full!\n");

return;

}

h->arr[h->size] = data;

int index = h->size;

h->size++;

while (index > 0) {

int parent = (index - 1) / 2;

if ((h->isMax && h->arr[index] > h->arr[parent]) ||

(!h->isMax && h->arr[index] < h->arr[parent])) {

int temp = h->arr[parent];

h->arr[parent] = h->arr[index];

h->arr[index] = temp;

index = parent;

} else {

break;

}

}

}

int extract(heap\* h) {

if (h->size == 0) {

printf("Heap is empty!\n");

return -1;

}

int extracted = h->arr[0];

h->arr[0] = h->arr[h->size - 1];

h->size--;

heapify(h, 0);

return extracted;

}

void printHeap(heap\* h) {

for (int i = 0; i < h->size; i++) printf("%d ", h->arr[i]);

printf("\n");

}

int main() {

int choice, heapType, value;

heap\* minHeap = createHeap(20, 0);

heap\* maxHeap = createHeap(20, 1);

while (1) {

printf("\nHeap Operations:\n");

printf("1. Insert into Min Heap\n");

printf("2. Insert into Max Heap\n");

printf("3. Extract from Min Heap\n");

printf("4. Extract from Max Heap\n");

printf("5. Display Min Heap\n");

printf("6. Display Max Heap\n");

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert into Min Heap: ");

scanf("%d", &value);

insert(minHeap, value);

break;

case 2:

printf("Enter value to insert into Max Heap: ");

scanf("%d", &value);

insert(maxHeap, value);

break;

case 3:

printf("Extracted from Min Heap: %d\n", extract(minHeap));

break;

case 4:

printf("Extracted from Max Heap: %d\n", extract(maxHeap));

break;

case 5:

printf("Min Heap: ");

printHeap(minHeap);

break;

case 6:

printf("Max Heap: ");

printHeap(maxHeap);

break;

case 7:

printf("Exiting...\n");

free(minHeap->arr);

free(maxHeap->arr);

free(minHeap);

free(maxHeap);

return 0;

default:

printf("Invalid choice! Please try again.\n");

}

}

return 0;

}

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 1

Enter value to insert into Min Heap: 2

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 1

Enter value to insert into Min Heap: 4

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 1

Enter value to insert into Min Heap: 6

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 1

Enter value to insert into Min Heap: 8

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 2

Enter value to insert into Max Heap: 3

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 2

Enter value to insert into Max Heap: 6

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 2

Enter value to insert into Max Heap: 9

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 2

Enter value to insert into Max Heap: 12

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 3

Extracted from Min Heap: 2

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 4

Extracted from Max Heap: 12

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 5

Min Heap: 4 8 6

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 6

Max Heap: 9 3 6

Heap Operations:

1. Insert into Min Heap

2. Insert into Max Heap

3. Extract from Min Heap

4. Extract from Max Heap

5. Display Min Heap

6. Display Max Heap

7. Exit

Enter your choice: 7

Exiting...