***// PRIORITY QUEUE USING HEAP***

#include <stdlib.h>

#include <stdio.h>

typedef struct heap {

int capacity;

int size;

int elements[1];

} heap;

heap\* heap\_construct(int initial\_capacity);

void heap\_destruct(heap\* heap);

int heap\_get\_capacity(heap \*heap);

int heap\_get\_size(heap \*heap);

*// Insert data into heap, return 1 if inserted, return 0 if size > capacity*

int heap\_insert(heap \*heap, int data);

*// Return the min value without deleting it*

int heap\_get\_min(heap \*heap);

*// Delete the min value in the heap*

int heap\_remove(heap \*heap);

*/\* Print the values(data) in ascending order by deleting the min value node from the heap each time\*/*

void heap\_print(heap \*heap);

int main() {

int choice, loop = 1;

int initial\_capacity = 0;

scanf("%d", &initial\_capacity);

heap \*my\_heap = heap\_construct(initial\_capacity);

while(loop) {

scanf("%d", &choice);

switch(choice) {

int number\_of\_elements, element, index;

int data;

case 1: *// Insert elements*

scanf("%d", &element);

if(!heap\_insert(my\_heap, element))

printf("OVERFLOW\n");

Break;

case 2: */\*Print elements in sorted order, and empties the heap. \*/*

heap\_print(my\_heap);

heap\_destruct(my\_heap);

loop = 0;

Break;

default: heap\_destruct(my\_heap);loop = 0;

}

}

return 0;

}

heap\* heap\_construct(int n)

{

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

temp->capacity=n;

temp->size=0;

\*(temp->elements)=(int\*)malloc(n\*sizeof(int));

for(int i=0;i<n;i++)

temp->elements[i]=0;

return temp;

}

void heap\_destruct(heap\* tmp)

{

}

int heap\_get\_capacity(heap\* tmp)

{

return tmp->capacity;

}

int heap\_get\_size(heap\* tmp)

{

return tmp->size;

}

int heap\_insert(heap\* tmp,int data)

{

int i=0;

while(tmp->elements[i]!=0 && i<tmp->capacity)

i++;

if(i<tmp->capacity)

{

tmp->elements[i]=data;

tmp->size++;

return 1;

}

else

return 0;

}

int heap\_get\_min(heap\* tmp)

{

int min=tmp->elements[0];

for(int i=1;i<tmp->size;i++)

{

if(tmp->elements[i]<min)

min=tmp->elements[i];

}

return min;

}

int heap\_remove(heap\* tmp)

{

int min=heap\_get\_min(tmp);

for(int i=0;i<tmp->size;i++)

{

if(tmp->elements[i]==min)

{

for(int k=i;k<tmp->size;k++)

tmp->elements[k]=tmp->elements[k+1];

tmp->size--;

return min;

}

}

}

void heap\_print(heap \*my\_heap)

{

while(my\_heap->size!=0)

{

int min=heap\_remove(my\_heap);

printf("%d\n",min);

}

}