SSN COLLEGE OF ENGINEERING (Autonomous)

Affiliated to Anna University

DEPARTMENT OF CSE

UCS308 Data Structures Lab

Assignment 10

Priority Queue Using Binary Heap

Register Number : 185001131

Name : Sai Charan B

Class : CSE – B

#Employee.h

```
typedef struct Employee{
	char name[30];
	int id;
	float salary;
}Employee;

Employee getEmployee(){
	Employee e;

	printf("Enter the name   : ");
	scanf("%[^\n]",e.name);
	printf("Enter the id     : ");
	scanf("%d",&e.id);
	printf("Enter the salary : ");
	scanf("%f",&e.salary);
	getchar();
	printf("\n");
	return e;
}

void putEmployee(const Employee e){
        printf("Name     : %s\n",e.name);
        printf("ID       : %d\n",e.id);
        printf("Salary   : %.2f\n",e.salary);
}
```

#MaxHeap.h

```
typedef Employee Data;

typedef struct PriorityQueue{
        int capacity;
        int size;
        Data* arr;
}PriorityQueue;

typedef PriorityQueue* PQueue;

int isFull(PQueue Q){
        return Q -> size == Q -> capacity;
}

int isEmpty(PQueue Q){
        return Q -> size == 0;
}


PQueue createPQueue(const int maxsize){
        PQueue tmp = (PQueue)malloc(sizeof(PriorityQueue));

        tmp -> capacity = maxsize;
        tmp -> size = 0;
        tmp -> arr = (Data*)malloc(sizeof(Data) * maxsize);

        tmp -> arr[0].salary = 999999.9;
        return tmp;
}

void enqueue(PQueue q,const Data d){
        if(isFull(q)){
                printf("Queue Full!\n");
                return;
        }
        int i = ++q -> size;
        for(; q -> arr[i/2].salary < d.salary ; i /= 2)
```

```
                q -> arr[i] = q -> arr[i/2];

        q -> arr[i] = d;

}

Data dequeue(PQueue q){
        if(isEmpty(q)){
                printf("Queue Empty!\n");
                return q -> arr[0];
        }
        int i,child;
        Data min,last;

        min = q -> arr[1];
        last = q -> arr[q -> size--];

        for(i = 1; i * 2 <= q -> size ; i = child){
                child = i * 2;

                if(child != q -> size && q -> arr[child + 1].salary > q ->
arr[child].salary)
                        child ++;
                if(last.salary < q -> arr[child].salary)
                        q -> arr[i] = q -> arr[child];
                else
                        break;
        }

        q -> arr[i] = last;
        return min;
}

void display(PQueue Q){
        for(int i = 1 ; i <= Q -> size ; i++)
                putEmployee(Q -> arr[i]);
}
```

#Main.c

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

#include "Employee.h"
```

```c
#include "MaxHeap.h"

int main(void){
	PQueue q = createPQueue(10);

	for(int i = 0 ; i < 5 ; i++){
		enqueue(q,getEmployee());
		printf("Queue after adding: \n");
		display(q);
		printf("------------------------\n");
	}


	printf("De-Queued Element\n");
	putEmployee(dequeue(q));

}
```

Output:

```
Enter the name : A

Enter the id : 1

Enter the salary : 50


Queue after adding:

Name : A

ID : 1

Salary : 50.00

------------------------

Enter the name : B

Enter the id : 2

Enter the salary : 70


Queue after adding:
```

```
Name : B

ID : 2

Salary : 70.00

Name : A

ID : 1

Salary : 50.00

------------------------

Enter the name : C

Enter the id : 3

Enter the salary : 40


Queue after adding:

Name : B

ID : 2

Salary : 70.00

Name : A

ID : 1

Salary : 50.00

Name : C

ID : 3

Salary : 40.00

------------------------

Enter the name : D

Enter the id : 4

Enter the salary : 700


Queue after adding:

Name : D

ID : 4

Salary : 700.00

Name : B

ID : 2

Salary : 70.00
```

```
Name : C

ID : 3

Salary : 40.00

Name : A

ID : 1

Salary : 50.00

------------------------

Enter the name : E

Enter the id : 5

Enter the salary : 300


Queue after adding:

Name : D

ID : 4

Salary : 700.00

Name : E

ID : 5

Salary : 300.00

Name : C

ID : 3

Salary : 40.00

Name : A

ID : 1

Salary : 50.00

Name : B

ID : 2

Salary : 70.00

------------------------

De-Queued Element

Name : D

ID : 4

Salary : 700.00
```