```
#include <iostream>
#include <string>
using namespace std;
class mobileUsers{
private:
    string name;
    long long mobileNo;
    double billAmount;
public:
    mobileUsers() : name(""), mobileNo(0), billAmount(0.0) {}
friend void classRecord(mobileUsers users[], int count);
double getBillAmount() const{
        return billAmount; }
    int getMobileNo() const { return mobileNo; }
void displayUser() const{
        cout << "Name: " << name << ", Mobile Number: " << mobileNo << ", Bill
Amount: $" << billAmount << endl;}};
void heapSort(mobileUsers users[], int count);
void heapify(mobileUsers users[], int count, int i);
void displayUsers(const mobileUsers users[], int count, const string &message = "");
void classRecord(mobileUsers users[], int count){
    for (int i = 0; i < count; i++){
        cout << "\nEnter details for User " << i + 1 << ":" << endl;
cout << "Enter Name: ";
        cin.ignore();
        getline(cin, users[i].name);
cout << "Enter Mobile Number: ";
        cin >> users[i].mobileNo;
cout << "Enter Bill Amount: ";
        cin >> users[i].billAmount;}
cout << "\nUser Details (Before Sorting):" << endl;
  displayUsers(users, count);
heapSort(users, count);
cout << "\nUser Details (After Sorting by Bill Amount):" << endl;
    displayUsers(users, count);}
void displayUsers(const mobileUsers users[], int count, const string &message){
    if (!message.empty()){
        cout << message << endl;}
    for (int i = 0; i < count; i++){
        cout << "User " << i + 1 << ": ";
        users[i].displayUser();}
    cout << endl;}
void heapify(mobileUsers users[], int count, int i){
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
if (left < count && users[left].getBillAmount() > users[largest].getBillAmount()){
        largest = left;}
if (right < count && users[right].getBillAmount() > users[largest].getBillAmount()){
        largest = right;}
if (largest != i){
        swap(users[i], users[largest]);
        heapify(users, count, largest);}}
void heapSort(mobileUsers users[], int count){
for (int i = count / 2 - 1; i >= 0; i--){
        heapify(users, count, i);}
    displayUsers(users, count, "After building the max heap:");
for (int i = count - 1; i > 0; i--){
        swap(users[0], users[i]);
        displayUsers(users, count, "After moving the largest element to the end:");
        heapify(users, i, 0);
        displayUsers(users, i, "After re-heapifying:");}}
int linearSearch(mobileUsers users[], int count, double target){
    for (int i = 0; i < count; i++){
        if (users[i].getBillAmount() == target){
            return i;}}return -1;}
```

```
int binarySearch(mobileUsers users[], int left, int right, double target){
    if (right >= left){
        int mid = left + (right - left) / 2;
if (users[mid].getBillAmount() == target){
            return mid;}
if (users[mid].getBillAmount() > target){
            return binarySearch(users, left, mid - 1, target);}
return binarySearch(users, mid + 1, right, target);}
return -1;}
int binarySearchNonRecursive(mobileUsers users[], int count, double target){
    int left = 0;
    int right = count - 1;
while (left <= right){
        int mid = left + (right - left) / 2;
if (users[mid].getBillAmount() == target){
            return mid;}
if (users[mid].getBillAmount() > target){
            right = mid - 1;}
        else{
            left = mid + 1;}}
return -1;}
int partition(mobileUsers arr[], int left, int right){
    mobileUsers pivot = arr[right];
    int i = (left - 1);
    for (int j = left; j <= right - 1; j++){
        if (arr[j].getMobileNo() <= pivot.getMobileNo()){
            i++;
            swap(arr[i], arr[j]);}}
    swap(arr[i + 1], arr[right]);
    return (i + 1);}
void quickSort(mobileUsers arr[], int left, int right){
    if (left < right){
        int pivotIndex = partition(arr, left, right);
        quickSort(arr, left, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, right);}}
int main(){
    int count;
    cout << "Enter the number of users: ";
    cin >> count;
    mobileUsers users[count];
    classRecord(users, count);
    // Linear Search
    double target;
    cout << "Enter the bill amount to search for: ";
    cin >> target;
    int index = linearSearch(users, count, target);
    if (index != -1){
        cout << "Bill Amount found at index " << index << endl;}
    else{
        cout << "Bill Amount not found" << endl;}
    // Binary Search
    index = binarySearch(users, 0, count - 1, target);
    if (index != -1){
        cout << "Bill Amount found at index " << index << endl;}
    else{
        cout << "Bill Amount not found" << endl;}
    // Binary Search Non-Recursive
    index = binarySearchNonRecursive(users, count, target);
    if (index != -1){
        cout << "Bill Amount found at index " << index << endl;}
    else{
        cout << "Bill Amount not found" << endl;}
    // Quick Sort
    quickSort(users, 0, count - 1);
   cout << "\nUser Details (After Sorting by Bill Amount using Quick Sort):" << endl;
    displayUsers(users, count);return 0;}
```