Đã bắt đầu vào lúc	Thứ sáu, 9 Tháng mười hai 2022, 7:35 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ sáu, 16 Tháng mười hai 2022, 4:26 PM
Thời gian thực hiện	6 ngày 20 giờ
Điểm	**8,51** của 10,00 (**85,1**%)

Câu hỏi 1

Chính xác

Điểm 0,81 của 1,00

Implement functions: **Peek, Pop, Size, Empty, Contains** to a maxHeap. If the function cannot execute, **return -1**.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <cmath>
#include <vector>
#include <algorithm>
using namespace std;
#define SEPARATOR "#<ab@17943918#@>#"
template<class T>
class Heap {
protected:
    T* elements;
    int capacity;
    int count;
public:
    Heap()
    {
        this->capacity = 10;
        this->count = 0;
        this->elements = new T[capacity];
    }
    ~Heap()
    {
        delete[]elements;
    }
    void push(T item);

    bool isEmpty();
    bool contains(T item);
    T peek();
    bool pop();
    int size();

    void printHeap()
    {
        cout << "Max Heap [ ";
        for (int i = 0; i < count; i++)
            cout << elements[i] << " ";
        cout << "]\n";
    }
private:
    void ensureCapacity(int minCapacity);
    void reheapUp(int position);
    void reheapDown(int position);
};
//Your code goes here
```

For example:

Test	Result

`Heap<int> maxHeap;`
`for (int i=0;i<10;i++){`
`  maxHeap.push(i);`
`}`
`cout << maxHeap.size();`
	10

`Heap<int> maxHeap;`
`for (int i=0;i<10;i++){`
`  maxHeap.push(i);`
`}`
`cout << maxHeap.isEmpty();`
	0

Answer: (penalty regime: 0, 0, 5, ... %)

Reset answer

```
template<class T>
int Heap<T>::size(){
    return count;
}

//template<class T>
//bool Heap<T>::isEmpty(){
//    return !count;
//}
template<class T>
bool Heap<T>::isEmpty(){
    if(count==0) return 1;
    return 0;
}
template<class T>
T Heap<T>::peek(){
    if(count !=0 )return elements[0];
    else return -1;
}

template<class T>
bool Heap<T>::contains(T item){
    for(int i=0;i<count;i++){
        if(elements[i]==item) return 1;
    }
    return 0;
}

template<class T>
bool Heap<T>::pop(){
    if(count==0) return -1;
    elements[0]=elements[count-1];
    count--;
    reheapDown(0);
    return 1;
}
```

	Test	Expected	Got	

✔	`Heap<int> maxHeap;`
	`for (int i=0;i<10;i++){`
	`  maxHeap.push(i);`
	`}`
	`cout << maxHeap.size();`
		10	10	✔

✔	`Heap<int> maxHeap;`
	`for (int i=0;i<10;i++){`
	`  maxHeap.push(i);`
	`}`
	`cout << maxHeap.isEmpty();`
		0	0	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00. Tính toán cho lần làm bài trước đó, điểm **0,81/1,00**.

Câu hỏi 2

Chính xác

Điểm 1,00 của 1,00

Implement function push to push a new item to a maxHeap. You also have to implement ensureCapacity and reheapUp to help you achieve that.

```
template
class Heap{
protected:
    T *elements;
    int capacity;
    int count;

public:
    Heap()
    {
        this->capacity = 10;
        this->count = 0;
        this->elements = new T[capacity];
    }
    ~Heap()
    {
        delete []elements;
    }
    void push(T item);
    void printHeap()
    {
        cout << "Max Heap [ ";
        for (int i = 0; i < count; i++)
            cout << elements[i] << " ";
        cout << "]";
    }

private:
    void ensureCapacity(int minCapacity);
    void reheapUp(int position);
};

// Your code here
```

For example:

Test	Result

`Heap<int> maxHeap;`
`for(int i = 0; i <5;i++)`
`  maxHeap.push(i);`
`maxHeap.printHeap();`
	`Max Heap [ 4 3 1 0 2 ]`

Answer: (penalty regime: 0, 0, 5, ... %)

Reset answer

```
template<class T>
void Heap<T>::push(T item){
    elements[count]=item;
    reheapUp(count);
    count++;
    if(count>=capacity) ensureCapacity(count*2);
}

template<class T>
void Heap<T>::ensureCapacity(int minCapacity){
    T* newele=new T[minCapacity];
    for(int i=0;i<capacity;i++){
        newele[i]=elements[i];
    }
    delete []elements;
    elements=newele;
    capacity= minCapacity;
}
```

```
19
20  template<class T>
21  void Heap<T>::reheapUp(int position){
22      while(position!=0){
23          int par=(position-1)/2;
24          if(elements[par]<elements[position]){
25              swap(elements[par],elements[position]);
26              position=par;
27          }
28          else break;
29      }
30  }
```

	Test	Expected	Got	

✔	`Heap<int> maxHeap;`
	`for(int i = 0; i <5;i++)`
	`  maxHeap.push(i);`
	`maxHeap.printHeap();`
		`Max Heap [ 4 3 1 0 2 ]`	`Max Heap [ 4 3 1 0 2 ]`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 3

Chính xác

Điểm 1,00 của 1,00

Given an array which the elements in it are random. Now we want to build a Max heap from this array. Implement functions Reheap up and Reheap down to heapify element at index position. We will use it to build a heap in next question.

To keep things simple, this question will separate the heap array, not store it in the class heap

```
void reheapDown(int maxHeap[], int numberOfElements, int index);
void reheapUp(int maxHeap[], int numberOfElements, int index);
```

For example:

Test	Result

int arr[] = {1,2,3,4,5,6,7,8};
int size = sizeof(arr)/sizeof(arr[0]);
reheapDown(arr,size,0);
cout << "[";
for(int i=0;i<size;i++)
  cout << arr[i] << " ";
cout << "]";
	[3 2 7 4 5 6 1 8]

int arr[] = {1,2,3,4,5,6,7,8};
int size = sizeof(arr)/sizeof(arr[0]);
reheapUp(arr,size,7);
cout << "[";
for(int i=0;i<size;i++)
  cout << arr[i] << " ";
cout << "]";
	[8 1 3 2 5 6 7 4]

Answer: (penalty regime: 0, 0, 5, ... %)

Reset answer

```
void reheapDown(int maxHeap[], int numberOfElements, int index)
{
    if(maxHeap==nullptr||index>numberOfElements) return ;
    for(int i=index;i<numberOfElements;){
        int L=2*i+1;
        int R=2*i+2;
        int max;
        if(L<numberOfElements&&maxHeap[i]<maxHeap[L]){
            max=L;
        }
        else max=i;
        if(R<numberOfElements&&maxHeap[i]<maxHeap[R]){
            max=R;
        }
        if(maxHeap[max]>maxHeap[i]){
            swap(maxHeap[max],maxHeap[i]);
            i=max;
        }
        else break;
    }
}

void reheapUp(int maxHeap[], int numberOfElements, int index)
{
    if(maxHeap==nullptr||index>numberOfElements) return ;
    for(int i=index;i>0;){
        int parent=(i-1)/2;
        if(maxHeap[i]>maxHeap[parent]){
            swap(maxHeap[parent],maxHeap[i]);
            i=parent;
        }
        else break;
    }
}
```

	Test	Expected	Got	

✔	`int arr[] = {1,2,3,4,5,6,7,8};`
	`int size = sizeof(arr)/sizeof(arr[0]);`
	`reheapDown(arr,size,0);`
	`cout << "[ ";`
	`for(int i=0;i<size;i++)`
	`  cout << arr[i] << " ";`
	`cout << "]";`
		[3 2 7 4 5 6 1 8]	[3 2 7 4 5 6 1 8]	✔

✔	`int arr[] = {1,2,3,4,5,6,7,8};`
	`int size = sizeof(arr)/sizeof(arr[0]);`
	`reheapUp(arr,size,7);`
	`cout << "[ ";`
	`for(int i=0;i<size;i++)`
	`  cout << arr[i] << " ";`
	`cout << "]";`
		[8 1 3 2 5 6 7 4]	[8 1 3 2 5 6 7 4]	✔

✔	`int arr[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};`
	`int size = sizeof(arr)/sizeof(arr[0]);`
	`reheapUp(arr,size,13);`
	`reheapUp(arr,size,12);`
	`cout << "[ ";`
	`for(int i=0;i<size;i++)`
	`  cout << arr[i] << " ";`
	`cout << "]";`
		[14 2 13 4 5 1 3 8 9 10 11 12 6 7 15]	[14 2 13 4 5 1 3 8 9 10 11 12 6 7 15]	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 4

Đúng một phần

Điểm 0,90 của 1,00

Implement method remove to **remove** the element with given value from a **maxHeap**, **clear** to remove all elements and bring the heap back to the initial state. You also have to implement method **getItem** to help you. Some given methods that you don't need to implement again are **push**, **printHeap**, **ensureCapacity**, **reheapUp**, **reheapDown**.

```
class Heap {
protected:
    T* elements;
    int capacity;
    int count;
public:
    Heap()
    {
        this->capacity = 10;
        this->count = 0;
        this->elements = new T[capacity];
    }
    ~Heap()
    {
        delete[]elements;
    }
    void push(T item);
    int getItem(T item);
    void remove(T item);
    void clear();
    void printHeap()
    {
        cout << "Max Heap [ ";
        for (int i = 0; i < count; i++)
            cout << elements[i] << " ";
        cout << "]\n";
    }
private:
    void ensureCapacity(int minCapacity);
    void reheapUp(int position);
    void reheapDown(int position);
};

// Your code here
```

For example:

Test	Result

`Heap<int> maxHeap;`
`int arr[] = {42,35,30,15,20,21,18,3,7,14};`
`for (int i = 0; i < 10; i++)`
`  maxHeap.push(arr[i]);`
`maxHeap.remove(42);`
`maxHeap.remove(35);`
`maxHeap.remove(30);`
`maxHeap.printHeap();`
	`Max Heap [ 21 20 18 15 14 7 3 ]`

`Heap<int> maxHeap;`
`int arr[] = {78, 67, 32, 56, 8, 23, 19, 45};`
`for (int i = 0; i < 8; i++)`
`  maxHeap.push(arr[i]);`
`maxHeap.remove(78);`
`maxHeap.printHeap();`
	`Max Heap [ 67 56 32 45 8 23 19 ]`

`Heap<int> maxHeap;`
`int arr[] = { 13, 19, 20, 7, 15, 12, 16, 10, 8, 9, 3, 6, 18, 2, 14, 1, 17, 4, 11, 5 };`
`for (int i = 0; i < 20; ++i)`
`  maxHeap.push(arr[i]);`
`maxHeap.clear();`
`maxHeap.printHeap();`
	`Max Heap [ ]`

Answer: (penalty regime: 10, 20, 30, ... %)

Reset answer

```
template<class T>
int Heap<T>::getItem(T item) {
    // TODO: return the index of item in heap
    for(int i=0; i<count; i++) {
        if(elements[i] == item) return i;
    }
    return -1;
}

template<class T>
void Heap<T>::remove(T item) {
    // TODO: remove the element with value equal to item
    int index = getItem(item);
    elements[index] = elements[--count];
    reheapUp(index);
    reheapDown(index);
}

template<class T>
void Heap<T>::clear() {
    // TODO: delete all elements in heap
    count = 0;
}
```

	Test	Expected	Got	

✔	`Heap<int> maxHeap;`
	`int arr[] = {42,35,30,15,20,21,18,3,7,14};`
	`for (int i = 0; i < 10; i++)`
	`  maxHeap.push(arr[i]);`
	`maxHeap.remove(42);`
	`maxHeap.remove(35);`
	`maxHeap.remove(30);`
	`maxHeap.printHeap();`
		`Max Heap [ 21 20 18 15 14 7 3 ]`	`Max Heap [ 21 20 18 15 14 7 3 ]`	✔

✔	`Heap<int> maxHeap;`
	`int arr[] = {78, 67, 32, 56, 8, 23, 19, 45};`
	`for (int i = 0; i < 8; i++)`
	`  maxHeap.push(arr[i]);`
	`maxHeap.remove(78);`
	`maxHeap.printHeap();`
		`Max Heap [ 67 56 32 45 8 23 19 ]`	`Max Heap [ 67 56 32 45 8 23 19 ]`	✔

✔	`Heap<int> maxHeap;`
	`int arr[] = { 13, 19, 20, 7, 15, 12, 16, 10, 8, 9, 3, 6, 18, 2, 14, 1, 17, 4, 11, 5 };`
	`for (int i = 0; i < 20; ++i)`
	`  maxHeap.push(arr[i]);`
	`maxHeap.clear();`
	`maxHeap.printHeap();`
		`Max Heap [ ]`	`Max Heap [ ]`	✔

Your code failed one or more hidden tests.

Đúng một phần

Điểm cho bài nộp này: 0,90/1,00.

Câu hỏi 5

Chính xác

Điểm 1,00 của 1,00

Your task is to implement heap sort (in ascending order) on an unsorted array.

```
#define SEPARATOR "#<ab@17943918#@>#"
#ifndef SORTING_H
#define SORTING_H
#include <iostream>
#include <queue>
using namespace std;
template <class T>
class Sorting {
public:
    /* Function to print an array */
    static void printArray(T *start, T *end)
    {
        long size = end - start;
        for (int i = 0; i < size - 1; i++)
            cout << start[i] << ", ";
        cout << start[size - 1];
        cout << endl;
    }

    //Helping functions go here
    static void heapSort(T* start, T* end){
        //TODO
        Sorting<T>::printArray(start,end);
    }

};
#endif /* SORTING_H */
```

For example:

Test	Result
int arr[4]={4,2,9,1}; Sorting<int>::heapSort(&arr[0],&arr[4]);	1, 2, 4, 9
int arr[4]={-1,0,2,3}; Sorting<int>::heapSort(&arr[0],&arr[4]);	-1, 0, 2, 3

Answer: (penalty regime: 0, 0, 5, ... %)

Reset answer

```
1  //Helping functions go here
2  static void heapSort(T* start, T* end){
3      //TODO
4      priority_queue<int,vector<int>,greater<int>> pq;
5      for(int i=0;i<end-start;i++){
6          pq.push(start[i]);
7      }
8      for(int i=0;i<end-start;i++){
9          start[i]=pq.top();
10         pq.pop();
11     }
12     Sorting<T>::printArray(start,end);
13 }
```

	Test	Expected	Got	
✔	`int arr[4]={4,2,9,1};` `Sorting<int>::heapSort(&arr[0],&arr[4]);`	`1, 2, 4, 9`	`1, 2, 4, 9`	✔
✔	`int arr[4]={-1,0,2,3};` `Sorting<int>::heapSort(&arr[0],&arr[4]);`	`-1, 0, 2, 3`	`-1, 0, 2, 3`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 6

Chính xác

Điểm 1,00 của 1,00

In a fast food restaurant, a customer is served by following the first-come, first-served rule. The manager wants to minimize the total waiting time of his customers. So he gets to decide who is served first, regardless of how sooner or later a person comes.

Different kinds of food take different amounts of time to cook. And he can't cook food for two customers at the same time, which means when he start cooking for customer A, he has to finish A 's order before cooking for customer B. For example, if there are 3 customers and they come at time 0, 1, 2 respectively, the time needed to cook their food is 3, 9, 6 respectively. If the manager uses first-come, first-served rule to serve his customer, the total waiting time will be 3 + 11 + 16 = 30. In case the manager serves his customer in order 1, 3, 2, the total waiting time will be 3 + 7 + 17 = 27.

Note: The manager does not know about the future orders.

In this question, you should implement function **minWaitingTime** to help the customer find minimum total waiting time to serve all his customers. You are also encouraged to use data structure **Heap** to complete this question. You can use your own code of **Heap**, or use functions related to Heap in library <algorithm>.

For example:

Test	Result

`int n = 3;`
`int arrvalTime[] = { 0, 1, 2 };`
`int completeTime[] = { 3, 9, 6 };`
 	
`cout << minWaitingTime(n, arrvalTime, completeTime);`
	27

`int n = 4;`
`int arrvalTime[] = { 0, 4, 2, 5 };`
`int completeTime[] = { 4, 2, 3, 4 };`
 	
`cout << minWaitingTime(n, arrvalTime, completeTime);`
	21

Answer: (penalty regime: 10, 20, 30, ... %)

Reset answer

```
#include <vector>
#include <algorithm>
bool mycmp(pair<int,int> a,pair<int,int>b){
    return a.first>b.first;
}
bool myothercmp(pair<int,int> a,pair<int,int>b){
    return a.second>b.second;//
}
//bool mycmp2(pair<int,int>a,pair<int,int>)
int minWaitingTime(int n, int arrvalTime[], int completeTime[]) {
    // YOUR CODE HERE
    vector<pair<int,int>>a;
    for(int i=0;i<n;i++){
        pair<int,int> temp;
        temp.first=arrvalTime[i];
        temp.second=completeTime[i];
        a.push_back(temp);
    }
    int time=0;
    int waitingtime=0;
    vector<pair<int,int>> v;
    make_heap(a.begin(),a.end(),mycmp);
    while(a.size()!=0){
        while(a.size()!=0&&time>=a[0].first){
            v.push_back(a[0]);
            push_heap(v.begin(), v.end(),myothercmp);
            pop_heap(a.begin(),a.end(),mycmp);
            a.pop_back();
        }
        if(v.size()==0) {
            v.push_back(a[0]);
            push_heap(v.begin(), v.end(),myothercmp);
            pop_heap(a.begin(),a.end(),mycmp);
            a.pop_back();
```

```
            time = v[0].first;
        }
        if(v.size()!=0){
            pair<int,int> temp=v[0];
            pop_heap(v.begin(),v.end(),myothercmp);
            v.pop_back();
            waitingtime+=temp.second+time-temp.first;
            time+=temp.second;
        }
    }
    for(int i=0;v.size()!=0;i++){
        pair<int,int> temp=v[0];
        pop_heap(v.begin(),v.end(),myothercmp);
        v.pop_back();
        waitingtime+=temp.second+time-temp.first;
        time+=temp.second;
    }
    return waitingtime;
}
```

	Test	Expected	Got	

✔	`int n = 3;`
	`int arrvalTime[] = { 0, 1, 2 };`
	`int completeTime[] = { 3, 9, 6 };`
	
	`cout << minWaitingTime(n, arrvalTime, completeTime);`
		27	27	✔

✔	`int n = 4;`
	`int arrvalTime[] = { 0, 4, 2, 5 };`
	`int completeTime[] = { 4, 2, 3, 4 };`
	
	`cout << minWaitingTime(n, arrvalTime, completeTime);`
		21	21	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 7

Chính xác

Điểm 0,60 của 1,00

Cho template của class PrinterQueue có 2 phương thức bắt buộc:

1. `addNewRequest(int priority, string fileName)`

Phương thức đầu tiên sẽ thêm 1 file vào danh sách hàng đợi của máy in (bao gồm độ ưu tiên và tên file). Test case sẽ có tối đa 100 file cùng lúc trong hàng đợi

2. `print()`

Phương thức thứ hai sẽ in tên file kèm xuống dòng và xóa nó ra khỏi hàng đợi. Nếu không có file nào trong hàng đợi, phương thức sẽ in ra "No file to print" kèm xuống dòng.

PrinterQueue tuân theo các quy tắc sau:

- fileName có độ ưu tiên cao nhất sẽ được in trước.
- Các fileName có cùng độ ưu tiên sẽ in theo thứ tự FIFO (First In First Out) order.

Nhiệm vụ của bạn là hiện thực class PrinterQueue thỏa mãn các yêu cầu dữ liệu trên

Lưu ý: Bạn có thể thay đổi mọi thứ, thêm thư viện cần thiết ngoại trừ thay đổi tên class, prototype của 2 public method bắt buộc.

Giải thích testcase 1: File goodbye.pdf có độ ưu tiên là 2 và được thêm vào sớm hơn file goodnight.pdf (độ ưu tiên = 2) nên sẽ được in trước, sau đó đến file goodnight.pdf và cuối cùng là hello.pdf có độ ưu tiên thấp nhất (1)

For example:

Test	Result

`PrinterQueue* myPrinterQueue = new PrinterQueue();`
`myPrinterQueue->addNewRequest(1, "hello.pdf");`
`myPrinterQueue->addNewRequest(2, "goodbye.pdf");`
`myPrinterQueue->addNewRequest(2, "goodnight.pdf");`
`myPrinterQueue->print();`
`myPrinterQueue->print();`
`myPrinterQueue->print();`
	`goodbye.pdf`
	`goodnight.pdf`
	`hello.pdf`

`PrinterQueue* myPrinterQueue = new PrinterQueue();`
`myPrinterQueue->addNewRequest(1, "hello.pdf");`
`myPrinterQueue->print();`
`myPrinterQueue->print();`
`myPrinterQueue->print();`
	`hello.pdf`
	`No file to print`
	`No file to print`

Answer: (penalty regime: 0, 0, 0, 100 %)

Reset answer

```
#include <queue>
#include <vector>
#include <iostream>
using namespace std;
struct huh {
    bool operator() (pair<int, string> a, pair<int, string> b) {
        return a.first < b.first;
    }
};
class PrinterQueue
{
public:
    // your attributes
    class Node{
        public:
        Node*next;
        int pri;
        string s;
        Node(Node*next,int pri,string s){
            this->next=next;
            this->pri=pri;
            this->s=s;
        }
    };
    Node*root;
    PrinterQueue(){
        root=nullptr;
```

```
    }
public:
    // your methods
    void addNewRequest(int priority, string fileName)
    {
        if(root==nullptr) {
            root=new Node(nullptr,priority,fileName);
            return ;
        }
        Node*temp=root;
        Node*prev=nullptr;
        while(temp!=nullptr&&temp->pri>=priority){
            prev=temp;
            temp=temp->next;
        }
        if(prev==nullptr){
            Node* newroot=new Node(nullptr,priority,fileName);
            newroot->next=root;
            root=newroot;
        }
        else{
            Node *newnode=new Node(prev->next,priority,fileName);
            prev->next=newnode;
        }
    }

    void print()
    {
        // your code here
        // After some logic code, you have to print fileName with endline
        //cout<<pq.size()<<endl;
        if (root==nullptr) {
            cout << "No file to print" << endl;
            return;
        }
        cout<<root->s<<endl;
        Node*temp=root->next;
        delete root;
        root=temp;
    }
};
```

	Test	Expected	Got	

✔	PrinterQueue* myPrinterQueue = new PrinterQueue();
	myPrinterQueue->addNewRequest(1, "hello.pdf");
	myPrinterQueue->addNewRequest(2, "goodbye.pdf");
	myPrinterQueue->addNewRequest(2, "goodnight.pdf");
	myPrinterQueue->print();
	myPrinterQueue->print();
	myPrinterQueue->print();
		goodbye.pdf
		goodnight.pdf
		hello.pdf
			goodbye.pdf
			goodnight.pdf
			hello.pdf
				✔

✔	PrinterQueue* myPrinterQueue = new PrinterQueue();
	myPrinterQueue->addNewRequest(1, "hello.pdf");
	myPrinterQueue->print();
	myPrinterQueue->print();
	myPrinterQueue->print();
		hello.pdf
		No file to print
		No file to print
			hello.pdf
			No file to print
			No file to print
				✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00. Tính toán cho lần làm bài trước đó, điểm **0,60/1,00**.

Câu hỏi 8

Chính xác

Điểm 1,00 của 1,00

Given an integer array `nums`. In each operation, two numbers are removed from array, and their sum is pushed back to the array. The cost of the operation is the sum of the two removed numbers.

Request: Implement function:

```
int reduceSum(vector<int>& nums);
```

Where `nums` is the mentioned array (the number of elements is between `1` and `100000`). This function returns the minimum sum of cost after repeating the operation until the array is reduced to one element.

Example:

Given the array: `[1, 2, 3, 4]`

In the first operation, `1` and `2` are removed, and their sum is pushed back to the array. The cost of the operation is `3`. After the first operation, the array is: `[3, 3, 4]`.

Repeat the process, the minimum total cost should be `19`.

Note:

In this exercise, the libraries `iostream, string, cstring, climits, utility, vector, list, stack, queue, map, unordered_map, set, unordered_set, functional, algorithm` has been included and `namespace std` are used. You can write helper functions and class. Importing other libraries is allowed, but not encouraged.

For example:

Test	Result
`vector<int> nums {1, 2, 3, 4};` `cout << reduceSum(nums);`	19

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
struct cmp{
    bool operator()(int a,int b){
        return a>b;
    }
};
int reduceSum(vector<int>& nums) {
    // STUDENT ANSWER
    priority_queue<int,vector<int>,cmp> pq;
    for(unsigned int i=0;i<nums.size();i++){
        pq.push(nums[i]);
    }
    int sum=0;
    //return 1;
    while(pq.size()!=1){
        int t1=pq.top();
        pq.pop();
        int t2=pq.top();
        pq.pop();
        sum=sum+t1+t2;
        pq.push(t1+t2);
    }
    return sum;

}
```

	Test	Expected	Got	
✔	vector<int> nums {1, 2, 3, 4}; cout << reduceSum(nums);	19	19	✔
✔	vector<int> nums {88}; cout << reduceSum(nums);	0	0	✔
✔	vector<int> nums {62, 77, 75, 46, 40, 8, 89, 37, 48, 50, 64, 30, 57, 95, 60, 18, 28, 43, 69, 67}; cout << reduceSum(nums);	4481	4481	✔
✔	vector<int> nums {35, 90, 40, 69, 29, 48, 28, 41, 65, 76, 48, 89, 88, 78, 36, 25, 32, 6, 90, 20, 78, 7, 76, 54, 38, 7, 63, 56, 29, 91, 78, 20, 92, 63, 26, 59, 51, 96, 36, 48, 20, 17, 73, 23, 2, 47, 62, 14, 48, 95, 65, 15, 74, 17, 94, 6, 1, 67, 13, 24, 26, 31, 70, 68, 81, 52, 11, 1, 51, 42, 70, 28, 58, 49, 14, 67, 39, 72, 44, 62, 37, 94, 93, 41, 40, 69, 39, 75, 40, 85, 67, 77, 11, 11, 28, 98, 0, 26, 85, 9}; cout << reduceSum(nums);	30932	30932	✔
✔	vector<int> nums {5, 54, 95, 56, 40, 84, 87, 20, 17, 61, 83, 92, 79, 22, 11, 79, 24, 6, 10, 28, 46, 42, 72, 32, 92, 34, 70, 74, 12, 21, 96, 29, 38, 83, 75, 56, 88, 5, 3, 14, 67, 41, 86, 45, 87, 30, 46, 71, 59, 90, 11, 9, 79, 58, 33, 0, 25, 54, 89, 41, 13, 77, 21, 37, 88, 3, 85, 60, 96, 97, 27, 9, 38, 29, 34, 67, 71, 80, 19, 19, 59, 3, 88, 42, 27, 53, 77, 89, 82, 45, 56, 53, 95, 70, 20, 96, 13, 4, 9, 81, 18, 85, 82, 26, 24, 57, 15, 20, 77, 30, 1, 91, 85, 15, 59, 92, 14, 82, 58, 31, 87, 68, 76, 9, 86, 11, 48, 38, 15, 67, 37, 42, 76, 58, 85, 80, 74, 49, 56, 76, 11, 44, 29, 77, 54, 80, 84, 74, 60, 10, 85, 91, 41, 6, 0, 54, 10, 18, 50, 19, 15, 20, 62, 11, 92, 83, 71, 80, 80, 77, 29, 85, 83, 24, 59, 67, 89, 38, 64, 77, 53, 66, 92, 92, 30, 57, 85, 78, 59, 9, 29, 24, 82, 19, 59, 1, 20, 55, 73, 64, 55, 68, 44, 41, 99, 41, 70, 77, 69, 19, 39, 4, 45, 15, 35, 86, 91, 43, 26, 12, 89, 5, 10, 67, 35, 64, 69, 35, 43, 77, 7, 66, 57, 3, 82, 51, 1, 98, 34, 17, 20, 37, 92, 99, 57, 96, 65, 22, 5, 5, 96, 4, 56, 29, 8, 91, 37, 82, 47, 17, 80, 86, 52, 14, 28, 82, 38, 0, 92, 76, 48, 93, 49, 94, 12, 90, 3, 48, 51, 39, 6, 67, 39, 43, 81, 15, 68, 97, 42, 50, 51, 27, 49, 26, 40, 62, 58, 25, 42, 83, 76, 19, 96, 72, 47, 8, 94, 78, 0, 46, 92, 9, 44, 45, 74, 32, 18, 89, 18, 51, 45, 32, 81, 90, 97, 73, 64, 64, 21, 40, 66, 40, 89, 11, 22, 51, 20, 56, 43, 78, 97, 39, 5, 97, 13, 71, 94, 16, 53, 20, 81, 44, 62, 12, 11, 77, 8, 74, 72, 96, 3, 82, 91, 92, 67, 35, 92, 17, 55, 61, 34, 82, 8, 7, 38, 62, 63, 96, 98, 53, 68, 80, 1, 65, 53, 25, 54, 69, 26, 83, 72, 0, 78, 69, 5, 33, 78, 16, 1, 90, 24, 41, 88, 95, 47, 70, 22, 56, 21, 61, 95, 13, 17, 9, 84, 61, 66, 97, 68, 25, 31, 50, 58, 10, 97, 63, 67, 42, 54, 15, 59, 14, 47, 58, 29, 64, 24, 84, 18, 0, 63, 58, 32, 75, 14, 51, 55, 12, 93, 2, 13, 17, 35, 68, 53, 89, 64, 85, 45, 68, 27, 22, 42, 34, 24, 69, 56, 85, 53, 40, 41, 32, 17, 99, 66, 68, 41, 85, 1, 86, 10, 37, 61, 61, 51, 97, 99, 30, 27, 39, 85, 13, 29, 5, 12, 17, 8, 78, 77, 95}; cout << reduceSum(nums);	219667	219667	✔
✔	vector<int> nums {41, 99, 96, 95, 59, 16, 25, 59, 45, 28, 28, 20, 77, 86, 19, 50, 17, 65, 32, 84, 22, 84, 32, 16, 25, 34, 22, 60, 29, 77, 37, 8, 32, 61, 50, 54, 13, 60, 12, 32, 6, 68, 68, 73, 67, 17, 7, 91, 78, 39, 45, 99, 97, 97, 13, 33, 73, 43, 51, 89, 4, 14, 28, 91, 33, 26, 88, 39, 51, 5, 72, 12, 35, 61, 78, 2, 70, 85, 82, 0, 7, 24, 22, 51, 48, 79, 58, 97, 70, 10, 32, 90, 97, 52, 42, 45, 18, 34, 89, 37, 97, 5, 88, 6, 10, 31, 6, 57, 59, 99, 44, 6, 10, 56, 2, 45, 59, 12, 71, 83, 45, 77, 92, 47, 89, 44, 13, 56, 26, 89, 90, 69, 55, 73, 21, 94, 74, 41, 55, 52, 55, 24, 32, 16, 61, 47, 50, 44, 22, 0, 65, 15, 20, 25, 81, 35, 74, 62, 56, 87, 29, 38, 51, 45, 13, 40, 66, 4, 4, 92, 54, 27, 60, 70, 27, 46, 94, 9, 54, 77, 17, 69, 60, 26, 89, 5, 71, 57, 98, 98, 17, 11, 32, 58, 87, 7, 40, 24, 0, 45, 38, 84, 54, 16, 60, 41, 80, 61, 83, 22, 92, 30, 73, 62, 47, 27, 43, 71, 27, 16, 20, 96, 56, 78, 1, 37, 7, 91, 18, 27, 2, 64, 17, 57, 64, 85, 58, 91, 56, 68, 27, 73, 3, 53, 28, 42, 63, 74, 1, 13, 37, 69, 15, 19, 71, 54, 5, 59, 74, 27, 84, 74, 93, 25, 13, 51, 19, 57, 1, 94, 22, 51, 61, 15, 77, 94, 44, 94, 79, 36, 89, 56, 12, 42, 9, 21, 67, 15, 80, 47, 45, 73, 87, 41, 1, 1, 98, 45, 76, 53, 59, 79, 87, 11, 87, 8, 67, 87, 27, 86, 48, 91, 58, 32, 89, 83, 77, 36, 91, 44, 91, 95, 99, 17, 58, 92, 29, 11, 77, 89, 83, 52, 43, 58, 35, 87, 68, 12, 48, 2, 67, 9, 90, 98, 64, 73, 27, 70, 71, 56, 96, 73, 4, 33, 41, 5, 44, 75, 86, 16, 48, 97, 11, 66, 85, 67, 89, 99, 72, 26, 17, 35, 34, 73, 68, 90, 55, 14, 79, 13, 67, 59, 9, 78, 49, 58, 63, 15, 43, 61, 87, 26, 17, 58, 89, 64, 65, 58, 13, 79, 59, 16, 65, 76, 18, 72, 18, 49, 19, 29, 16, 43, 49, 8, 0, 80, 88, 74, 84, 50, 98, 71, 93, 59, 14, 64, 4, 44, 11, 70, 49, 4, 61, 20, 37, 75, 54, 23, 21, 66, 36, 92, 39, 93, 21, 18, 41, 78, 19, 52, 45, 79, 4, 49, 67, 47, 41, 6, 1, 37, 99, 64, 38, 29, 38, 11, 72, 47, 62, 94, 61, 78, 90, 16, 36, 17, 67, 26, 85, 48, 69, 70, 13, 25, 39, 23, 14, 64, 29, 50, 79, 60, 89, 20, 3, 19, 51, 33, 63, 1, 74, 9, 29, 99, 30, 49, 54, 37, 23, 56, 88, 78, 29, 2, 76, 12, 94, 41, 14, 32, 74, 94, 31, 99, 25, 70, 2, 84, 26, 33, 98, 4, 42, 76, 26, 23, 66, 74, 30, 1, 37, 84, 45, 69, 93, 33, 84, 25, 77, 38, 86, 14, 39, 43, 72, 65, 96, 34, 71, 14, 88, 93, 16, 88, 30, 54, 23, 54, 11, 17, 91, 16, 10, 69, 29, 32, 20, 47, 7, 38, 44, 4, 91, 23, 54, 69, 75, 98, 8, 15, 83, 9, 30, 67, 18, 45, 54, 60, 5, 6, 46, 64, 22, 18, 20, 65, 19, 24, 66, 31, 5, 30, 35, 45, 64, 46, 46, 15, 72, 31, 18, 75, 90, 15, 70, 44, 95, 96, 78, 94, 24, 41, 45, 80, 98, 7, 32, 2, 0, 44, 19, 43, 99, 30, 71, 90, 14, 47, 7, 59, 14, 27, 63, 24, 79, 54, 2, 16, 28, 56, 78, 12, 43, 31, 18, 93, 20, 44, 59, 5, 32, 4, 35, 9, 65, 4, 22, 50, 88, 38, 29, 19, 82, 10, 29, 27, 47, 94, 76, 79, 88, 70, 87, 54, 98, 58, 7, 38, 44, 80, 59, 82, 82, 96, 34, 94, 70, 75, 73, 31, 11, 44, 4, 44, 43, 33, 86, 36, 65, 81, 82, 64, 4, 72, 43, 79, 99, 84, 18, 85, 85, 35, 46, 93, 90, 21, 68, 66, 71, 63, 61, 15, 46, 96, 57, 58, 90, 7, 53, 22, 46, 41, 3, 9, 95, 32, 43, 36, 63, 85, 58, 8, 11, 50, 43, 53, 5, 26, 58, 93, 83, 81, 68, 79, 98, 75, 6, 53, 74, 8, 93, 42, 5, 9, 90, 48, 6, 78, 60, 57, 49, 7, 98, 16, 56, 0, 25, 31, 51, 36, 18, 51, 79, 36, 28, 15, 11, 67, 40, 56, 68, 41, 92, 9, 71, 98, 29, 26, 37, 64, 52, 30, 55, 16, 44, 89, 1, 17, 50, 79, 23, 53, 65, 79, 54, 68, 55, 3, 4, 46, 16, 64, 4, 69, 13, 85, 76, 67, 67, 19, 47, 8, 25, 27, 85, 51, 63, 11, 93, 29, 57, 5, 90, 27, 84, 58, 61, 6, 15, 43, 70, 88, 48, 14, 37, 16, 72, 86, 95, 52, 20, 74, 49, 90, 34, 7, 56, 33, 59, 33, 30, 61, 34, 69, 32, 82, 17, 72, 80, 9, 19, 54, 76, 51, 58, 7, 12, 6, 29, 18, 41, 70, 0, 45, 18, 48, 82, 52, 52, 57, 64, 45, 4, 64, 85, 18, 31, 20, 82, 73, 10, 33, 62, 41, 63, 5, 28, 86, 0, 20, 64, 3, 69, 79, 87, 50, 51, 20, 59, 24, 36, 31, 46, 5, 64, 19, 62, 18, 82, 27, 21, 81, 75, 26, 22, 64, 46, 66, 8, 62, 94, 1, 69, 51, 76, 17, 25, 46, 0, 33, 96, 44, 8, 43, 38, 42, 50, 97, 69, 23, 15, 95, 48, 23, 29, 1, 94, 35, 46, 56}; cout << reduceSum(nums);	468321	468321	✔

	Test	Expected	Got	
✔	vector<int> nums {45, 46, 45, 22, 24, 92, 97, 0, 95, 93, 76, 80, 81, 52, 78, 90, 2, 88, 21, 70, 54, 6, 71, 62, 38, 28, 73, 62, 18, 86, 59, 81, 54, 23, 62, 81, 17, 75, 77, 23, 67, 22, 83, 80, 78, 3, 95, 83, 3, 78, 1, 34, 86, 91, 65, 10, 99, 92, 42, 7, 86, 0, 56, 56, 81, 45, 76, 9, 25, 67, 67, 58, 54, 67, 78, 13, 71, 10, 18, 74, 69, 50, 68, 82, 11, 25, 50, 70, 75, 51, 22, 66, 38, 87, 14, 74, 37, 36, 12, 59, 70, 94, 24, 53, 29, 94, 13, 23, 56, 6, 9, 96, 29, 10, 88, 33, 94, 74, 89, 79, 70, 77, 89, 30, 30, 33, 16, 6, 2, 56, 50, 17, 89, 30, 80, 73, 98, 80, 27, 47, 81, 41, 40, 28, 9, 39, 53, 3, 84, 56, 51, 88, 24, 9, 16, 15, 98, 51, 23, 36, 16, 32, 25, 30, 7, 68, 10, 57, 67, 97, 17, 16, 42, 51, 56, 22, 3, 81, 48, 69, 39, 8, 11, 93, 9, 93, 95, 34, 99, 4, 98, 56, 55, 98, 2, 2, 90, 60, 59, 21, 38, 93, 6, 91, 52, 30, 45, 15, 2, 55, 12, 2, 41, 90, 22, 34, 29, 7, 41, 30, 99, 78, 50, 75, 95, 88, 64, 38, 60, 38, 14, 30, 17, 15, 39, 58, 1, 53, 66, 82, 61, 15, 42, 34, 48, 13, 40, 64, 52, 36, 30, 36, 51, 43, 58, 40, 40, 42, 20, 33, 83, 57, 4, 91, 71, 17, 52, 3, 67, 51, 52, 47, 38, 51, 62, 26, 2, 75, 96, 46, 14, 67, 60, 82, 24, 88, 43, 84, 99, 34, 5, 79, 77, 27, 65, 56, 35, 85, 40, 70, 8, 73, 78, 79, 88, 88, 79, 33, 30, 26, 31, 89, 15, 90, 53, 30, 96, 57, 1, 9, 6, 43, 76, 96, 41, 6, 4, 89, 43, 12, 51, 29, 66, 91, 61, 57, 99, 90, 34, 7, 36, 71, 56, 75, 24, 63, 21, 64, 37, 14, 99, 37, 34, 16, 81, 65, 23, 75, 73, 99, 94, 58, 63, 50, 27, 92, 88, 37, 55, 57, 50, 92, 43, 34, 60, 74, 35, 83, 1, 98, 25, 75, 82, 64, 52, 70, 91, 29, 87, 2, 61, 73, 52, 22, 5, 28, 1, 22, 56, 87, 44, 25, 80, 97, 25, 91, 2, 26, 82, 50, 88, 10, 90, 7, 97, 28, 58, 29, 3, 93, 77, 6, 68, 76, 13, 70, 68, 52, 39, 75, 37, 24, 14, 65, 32, 54, 1, 5, 51, 49, 27, 32, 68, 3, 11, 3, 73, 19, 90, 13, 36, 20, 25, 61, 29, 83, 9, 40, 65, 9, 76, 0, 74, 3, 17, 86, 1, 93, 7, 94, 15, 13, 32, 69, 78, 78, 16, 35, 69, 33, 79, 85, 67, 38, 71, 61, 93, 82, 0, 5, 35, 11, 2, 33, 20, 97, 1, 68, 79, 40, 64, 60, 77, 93, 92, 68, 88, 62, 13, 35, 62, 12, 56, 75, 31, 3, 11, 51, 11, 41, 79, 77, 36, 67, 80, 30, 98, 84, 14, 59, 11, 14, 34, 92, 93, 10, 66, 75, 54, 75, 30, 83, 96, 42, 19, 49, 53, 32, 53, 63, 12, 69, 40, 99, 88, 44, 30, 99, 77, 26, 68, 10, 37, 59, 38, 33, 97, 31, 48, 4, 24, 1, 39, 14, 90, 90, 63, 76, 80, 68, 53, 84, 96, 21, 8, 25, 1, 42, 16, 67, 96, 27, 12, 43, 80, 56, 72, 13, 22, 68, 93, 7, 32, 21, 32, 18, 73, 65, 84, 64, 90, 52, 20, 88, 2, 33, 8, 5, 40, 53, 86, 83, 36, 34, 20, 48, 29, 4, 53, 14, 83, 10, 61, 6, 77, 38, 81, 33, 12, 22, 10, 20, 40, 82, 40, 97, 82, 14, 26, 96, 63, 30, 36, 15, 85, 43, 6, 54, 24, 3, 7, 70, 66, 55, 40, 24, 9, 57, 6, 56, 48, 61, 1, 26, 56, 92, 19, 24, 25, 23, 88, 58, 18, 56, 15, 51, 6, 90, 76, 57, 31, 14, 84, 71, 95, 54, 50, 88, 72, 37, 11, 66, 64, 58, 13, 96, 14, 2, 43, 1, 27, 60, 12, 51, 9, 39, 81, 31, 67, 96, 40, 93, 9, 56, 42, 89, 39, 60, 47, 95, 68, 70, 85, 33, 85, 70, 6, 83, 49, 66, 82, 80, 60, 38, 88, 42, 99, 20, 11, 82, 14, 51, 37, 29, 36, 13, 27, 90, 6, 43, 15, 81, 50, 8, 58, 75, 38, 64, 36, 7, 22, 62, 86, 65, 66, 48, 99, 33, 40, 98, 34, 85, 28, 98, 74, 74, 85, 62, 61, 17, 43, 82, 75, 46, 37, 0, 83, 33, 96, 21, 60, 96, 81, 2, 89, 47, 0, 58, 28, 94, 9, 66, 56, 40, 36, 86, 57, 72, 51, 10, 22, 41, 6, 69, 41, 65, 66, 53, 93, 67, 71, 59, 96, 41, 69, 52, 74, 35, 52, 85, 29, 34, 42, 14, 35, 67, 67, 13, 89, 67, 87, 89, 33, 47, 93, 9, 30, 16, 68, 3, 2, 57, 14, 52, 33, 76, 82, 78, 5, 97, 58, 84, 81, 64, 94, 60, 9, 6, 64, 18, 38, 11, 68, 96, 98, 43, 38, 13, 27, 41, 61, 83, 8, 3, 86, 44, 24, 35, 9, 39, 30, 13, 23, 27, 68, 66, 50, 15, 36, 48, 5, 54, 76, 88, 63, 86, 41, 56, 53, 9, 42, 25, 57, 84, 58, 56, 18, 25, 19, 55, 30, 56, 45, 54, 95, 62, 34, 90, 13, 88, 51, 75, 42, 52, 35, 21, 63, 11, 66, 68, 91, 34, 57, 88, 89, 62, 80, 57, 87, 33, 36, 75, 48, 12, 94, 78, 97, 82, 95, 90, 16, 78, 41, 85, 73, 84, 15, 35, 49, 51, 57, 9, 7, 35, 51, 49, 53, 65, 17, 98, 72, 16, 89, 9, 56, 83, 5, 27, 58, 58, 59, 56, 10, 49, 76, 95, 95, 6, 48, 94, 93, 47, 50, 0, 38, 8, 67, 70, 60, 6, 31, 91, ...snip... , 94, 70, 35, 1, 7, 93, 28, 6, 47, 66, 72, 20, 38, 24, 40, 64, 13, 68, 24, 72, 24, 46, 16, 4, 32, 17, 76, 8, 28, 35, 1, 15, 42, 29, 10, 28, 21, 93, 59, 90, 12, 61, 28, 92, 97, 55, 58, 3, 43, 93, 22, 47, 13, 51, 86, 68, 1, 14, 8, 92, 64, 55, 59, 5, 70, 77, 92, 77, 0, 64, 30, 62, 52, 94, 46, 43, 40, 66, 54, 62, 3, 58, 11, 88, 49, 21, 92, 94, 93, 98, 5, 1, 96, 16, 28, 16, 7, 18, 58, 48, 33, 37, 69, 64, 90, 51, 33, 99, 15, 82, 57, 91, 3, 86, 63, 71, 26, 27, 50, 27, 36, 14, 27, 18, 64, 29, 6, 44, 53, 18, 7, 94, 60, 20, 49, 74, 55, 36, 97, 3, 18, 31, 34, 37, 28, 15, 79, 57, 93, 43, 57, 11, 93, 30, 59, 43, 91, 75, 71, 53, 7, 94, 40, 81, 5, 27, 17, 35, 47, 48, 71, 79, 0, 16, 1, 11, 66, 77, 86, 16, 21, 21, 80, 87, 98, 37, 16, 24, 59, 14, 75, 8, 43, 61, 79, 22, 47, 11, 37, 55, 66, 40, 81, 67, 3, 81, 50, 27, 37, 65, 41, 41, 64, 54, 8, 61, 17, 20, 7, 93, 62, 73, 85, 99, 67, 84, 46, 18, 20, 3, 13, 38, 84, 92, 54, 23, 81, 80, 90, 96, 69, 91, 26, 5, 64, 74, 31, 74, 27, 18, 35, 31, 53, 92, 9, 56, 24, 90, 89, 79, 42, 37, 30, 60, 40, 39, 39, 52, 47, 34, 0, 26, 89, 58, 23, 58, 18, 52, 59, 99, 68, 96, 32, 32, 36, 16, 39, 37, 94, 61, 18, 71, 50, 89, 51, 57, 80, 73, 50, 72, 64, 78, 9, 25, 99, 95, 72, 74, 6, 69, 44, 39, 20, 9, 26, 41, 2, 33, 39, 36, 29, 37, 71, 0, 36, 12, 49, 33, 50, 12, 26, 88, 50, 50, 56, 38, 65, 21, 39, 3, 96, 80, 12, 29, 30, 70, 70, 86, 29, 34, 22, 40, 80, 83, 16, 32, 79, 94, 43, 24, 99, 29, 3, 85, 43, 66, 25, 67, 60, 45, 58, 15, 43, 78, 73, 7, 20, 46, 15, 44, 79, 88, 44, 72, 72, 87, 75, 37, 77, 70, 29, 66, 50, 59, 46, 61, 32, 11, 44, 53, 59, 66, 52, 24, 89, 66, 46, 40, 40, 35, 5, 37, 37, 34, 39, 90, 33, 60, 10, 45, 12, 9, 15, 60, 25, 19, 46, 34, 61, 49, 22, 0, 8, 77, 74, 32, 60, 53, 28, 28, 19, 82, 81, 75, 71, 57, 21, 45, 91, 7, 0, 59, 57, 58, 54, 36, 82, 60, 55, 2, 26, 47, 74, 13, 85, 26, 39, 86, 5, 23, 94, 70, 49, 33, 74, 12, 72, 44, 13, 93, 46, 51, 94, 71, 36, 6, 45, 28, 87, 48, 58, 79, 11, 50, 32, 58, 11, 16, 55, 50, 88, 99, 64, 64, 61, 52, 48, 22, 50, 92, 16, 82, 34, 45, 95, 17, 57, 12, 96, 34, 70, 79, 62, 27, 83, 73, 37, 41, 38, 44, 95, 36, 32, 46, 70, 73, 25, 62, 92, 24, 67, 73, 13, 32, 26, 96, 7, 53, 19, 77, 28, 16, 99, 6, 71, 28, 99, 73, 40, 24, 26, 29, 89, 40, 75, 35, 21, 62, 43, 8, 93, 90, 72, 62, 92, 27, 99, 37, 27, 79, 4, 45, 40, 68, 16, 1, 17, 73, 3, 17, 41, 9, 98, 16, 0, 38, 79, 79, 72, 57, 88, 30, 63, 35, 1, 80, 93, 47, 97, 56, 9, 83, 14, 31, 8, 88, 27, 99, 97, 6, 23, 42, 81, 95, 88, 72, 13, 58, 21, 97, 24, 51, 4, 87, 10, 29, 42, 18, 29, 40, 4, 0, 43, 65, 74, 91, 21, 67, 46, 89, 99, 19, 39, 28, 47, 77, 26, 22, 14, 40, 69, 50, 84, 72, 38, 31, 3, 23, 3, 9, 59, 55, 3, 38, 41, 73, 91, 22, 77, 65, 19, 50, 37, 52, 54, 94, 3, 53, 85, 49, 46, 77, 43, 5, 44, 13, 8, 39, 41, 81, 47, 92, 55, 34, 84, 65, 78, 8, 31, 64, 59, 37, 99, 23, 25, 65, 58, 68, 61, 50, 69, 58, 49, 7, 78, 40, 68, 60, 96, 43, 17, 52, 87, 84, 66, 59, 1, 34, 26, 16, 72, 29, 16, 77, 5, 0, 2, 94, 89, 52, 74, 64, 22, 81, 92, 19, 35, 32, 5, 74, 95, 34, 82, 98, 27, 73, 51, 36, 19, 84, 81, 94, 52, 99, 22, 72, 31, 36, 68, 95, 76, 19, 77, 19, 23, 37, 2, 86, 94, 39, 42, 76, 97, 93, 52, 32, 51, 18, 50, 79, 66, 52, 72, 55, 57, 37, 52, 78, 67, 71, 56, 5, 34, 20, 47, 43, 53, 7, 71, 58, 51, 61, 15, 85, 30, 91, 39, 82, 3, 23, 80, 58, 78, 58, 18, 25, 31, 77, 31, 16, 14, 42, 53, 42, 29, 45, 71, 24, 32, 22, 52, 73, 58, 97, 68, 88, 25, 25, 12, 93, 74, 1, 36, 83, 90, 85, 3, 50, 0, 91, 93, 68, 36, 37, 48, 43, 83, 22, 85, 23, 48, 79, 56, 82, 55, 96, 62, 39, 20, 47, 74, 55, 13, 32, 22, 93, 43, 1, 72, 21, 47, 97, 21, 23, 73, 39, 51, 57, 1, 24, 33, 90, 93, 25, 91, 84, 29, 74, 65, 89, 17, 0, 49, 13, 20, 43, 91, 59, 82, 40, 95, 78, 39, 48, 36, 97, 60, 21, 72, 82, 11, 63, 8, 82, 94, 69, 30, 97, 61, 58, 55, 86, 93, 85, 62, 56, 47, 99, 6, 19, 48, 83, 75, 80, 29, 75, 44, 7, 11, 11, 51, 36, 93, 90, 41, 88, 85, 43, 68, 84, 79, 48, 26, 46, 23, 61, 64, 23, 2, 72, 39, 76, 64, 77, 5, 65, 57, 47, 24, 99, 4, 28, 14, 81, 79, 98, 48, 58, 64, 66, 65, 93, 3, 17, 49}; cout << reduceSum(nums);	2971358	2971358	✔

	Test	Expected	Got	
✔	vector<int> nums {84, 25, 8, 31, 69, 76, 9, 10, 66, 45, 20, 43, 56, 87, 11, 4, 91, 20, 52, 23, 99, 56, 54, 86, 10, 4, 87, 9, 52, 1, 13, 5, 83, 13, 36, 40, 32, 51, 62, 16, 21, 46, 4, 40, 57, 94, 49, 85, 25, 47, 16, 33, 49, 29, 31, 90, 62, 77, 73, 81, 41, 96, 59, 53, 20, 45, 71, 57, 99, 89, 28, 0, 41, 28, 10, 9, 52, 47, 53, 86, 51, 40, 80, 93, 11, 90, 80, 76, 96, 94, 21, 60, 20, 65, 16, 97, 72, 7, 1, 81, 38, 79, 75, 47, 94, 87, 46, 25, 77, 35, 56, 63, 44, 89, 91, 14, 9, 19, 86, 1, 26, 37, 78, 83, 65, 72, 87, 53, 51, 28, 39, 1, 1, 57, 10, 88, 77, 87, 39, 24, 85, 91, 34, 24, 47, 92, 58, 32, 55, 48, 49, 4, 2, 88, 57, 16, 68, 65, 17, 50, 65, 71, 53, 71, 20, 50, 18, 99, 84, 96, 45, 2, 38, 14, 98, 18, 70, 78, 76, 83, 37, 60, 36, 65, 35, 32, 53, 22, 38, 80, 90, 59, 59, 70, 34, 70, 11, 36, 23, 68, 80, 17, 74, 52, 21, 47, 37, 41, 20, 83, 34, 28, 63, 29, 91, 45, 37, 61, 91, 90, 29, 40, 29, 64, 71, 59, 66, 11, 38, 45, 19, 28, 26, 27, 60, 93, 98, 21, 78, 63, 73, 11, 44, 90, 89, 77, 0, 14, 52, 66, 51, 38, 14, 87, 34, 94, 75, 79, 23, 60, 49, 30, 49, 22, 48, 70, 63, 9, 50, 57, 26, 70, 65, 1, 41, 22, 19, 85, 20, 91, 56, 77, 36, 9, 82, 47, 79, 88, 9, 38, 2, 20, 72, 27, 55, 6, 40, 26, 32, 9, 3, 91, 57, 57, 36, 34, 85, 67, 73, 80, 86, 1, 93, 60, 43, 47, 67, 5, 9, 30, 11, 16, 26, 80, 19, 95, 90, 13, 24, 47, 74, 96, 71, 83, 86, 64, 22, 23, 93, 1, 56, 16, 68, 18, 67, 16, 55, 53, 91, 28, 79, 85, 51, 28, 5, 98, 19, 76, 49, 54, 94, 84, 86, 38, 45, 77, 5, 12, 59, 81, 5, 92, 50, 27, 14, 25, 22, 17, 17, 90, 21, 70, 50, 92, 31, 59, 11, 89, 10, 36, 33, 53, 37, 63, 24, 96, 68, 4, 47, 28, 84, 99, 88, 5, 56, 6, 45, 50, 95, 7, 60, 50, 72, 5, 25, 71, 18, 98, 69, 97, 22, 56, 13, 10, 9, 42, 99, 57, 66, 50, 84, 21, 95, 0, 95, 31, 59, 29, 57, 2, 56, 93, 77, 85, 78, 62, 78, 19, 48, 35, 31, 31, 25, 8, 42, 38, 30, 18, 80, 5, 46, 49, 27, 77, 93, 75, 74, 22, 21, 92, 85, 32, 10, 60, 16, 44, 22, 53, 93, 78, 17, 98, 4, 54, 77, 79, 39, 28, 19, 22, 94, 88, 28, 94, 65, 62, 43, 53, 27, 95, 92, 79, 38, 68, 26, 24, 80, 70, 64, 14, 95, 35, 57, 53, 21, 98, 16, 15, 37, 49, 30, 28, 26, 43, 20, 60, 57, 52, 75, 2, 49, 93, 67, 57, 67, 87, 38, 73, 66, 0, 20, 97, 48, 87, 0, 98, 5, 52, 64, 10, 72, 48, 46, 70, 16, 73, 74, 45, 72, 97, 55, 90, 58, 52, 90, 83, 29, 40, 48, 31, 93, 65, 18, 41, 23, 12, 6, 75, 1, 34, 56, 81, 87, 49, 52, 61, 95, 75, 94, 67, 54, 99, 73, 57, 95, 4, 24, 1, 87, 91, 87, 58, 74, 61, 66, 75, 33, 97, 39, 86, 48, 59, 12, 96, 35, 88, 26, 9, 19, 72, 64, 59, 50, 70, 63, 42, 27, 46, 11, 71, 50, 17, 54, 85, 40, 40, 60, 44, 98, 74, 59, 75, 39, 32, 32, 43, 75, 82, 85, 66, 69, 16, 52, 13, 17, 60, 80, 26, 11, 97, 10, 9, 94, 5, 79, 45, 52, 55, 40, 13, 76, 56, 58, 92, 94, 59, 32, 66, 40, 22, 77, 16, 49, 55, 74, 5, 18, 39, 52, 89, 58, 10, 97, 82, 74, 51, 25, 6, 57, 49, 99, 69, 38, 30, 61, 84, 88, 3, 12, 24, 69, 13, 79, 78, 18, 66, 30, 63, 94, 18, 2, 83, 9, 60, 60, 48, 50, 68, 20, 44, 97, 95, 7, 51, 90, 97, 26, 38, 22, 57, 20, 34, 11, 37, 69, 45, 61, 84, 33, 2, 20, 86, 56, 64, 23, 23, 39, 13, 92, 74, 60, 67, 33, 95, 21, 50, 81, 67, 43, 43, 51, 39, 52, 28, 21, 88, 25, 77, 90, 73, 45, 91, 17, 42, 69, 19, 89, 31, 55, 70, 37, 40, 37, 86, 87, 43, 14, 11, 2, 98, 66, 59, 56, 68, 20, 30, 61, 38, 21, 74, 67, 30, 48, 18, 12, 2, 93, 46, 47, 49, 78, 2, 72, 71, 75, 87, 71, 31, 68, 2, 13, 26, 17, 14, 44, 17, 89, 12, 32, 69, 35, 0, 51, 74, 68, 27, 70, 86, 53, 39, 32, 68, 61, 43, 19, 29, 66, 52, 20, 77, 57, 71, 24, 61, 82, 39, 76, 60, 65, 1, 12, 11, 37, 94, 56, 53, 15, 10, 35, 31, 99, 67, 69, 4, 73, 11, 6, 77, 34, 69, 59, 95, 64, 89, 41, 97, 47, 39, 72, 96, 41, 78, 53, 3, 69, 62, 55, 80, 48, 82, 54, 84, 48, 66, 37, 6, 28, 15, 74, 27, 67, 32, 8, 60, 20, 18, 84, 1, 31, 83, 75, 85, 62, 43, 10, 98, 70, 14, 39, 94, 42, 44, 48, 70, 52, 75, 77, 17, 96, 15, 99, 72, 54, 36, 70, 41, 43, 64, 46, 54, 14, 12, 53, 20, 50, 16, 84, 55, 8, 95, 65, 27, 55, 84, 80, 23, 96, 40, 84, 5, 23, 91, 0, 79, 13, 44, 45, 37, 32, 60, 95, 15, 48, 73, 63, 80, 11, 5, 15, 82, 87, 56, 72, 76, 58, 42, 23, 93, 26, 3, 79, 35, 75, 75, 6 ...snip... 9, 33, 36, 13, 42, 80, 15, 69, 55, 51, 45, 85, 87, 39, 46, 4, 95, 86, 16, 37, 57, 90, 90, 68, 83, 26, 12, 25, 16, 49, 19, 39, 52, 59, 57, 53, 82, 67, 98, 85, 55, 69, 44, 22, 8, 75, 4, 32, 42, 63, 30, 13, 9, 35, 7, 18, 47, 65, 17, 62, 95, 28, 51, 94, 66, 95, 6, 30, 60, 8, 59, 71, 10, 59, 13, 42, 3, 44, 33, 40, 16, 22, 41, 50, 44, 30, 30, 79, 52, 79, 53, 22, 76, 59, 75, 30, 86, 63, 40, 37, 58, 22, 33, 72, 46, 72, 53, 65, 7, 36, 83, 49, 89, 66, 24, 84, 43, 34, 74, 66, 13, 89, 92, 55, 69, 59, 68, 86, 25, 20, 18, 25, 52, 79, 82, 72, 98, 91, 69, 36, 12, 60, 84, 14, 43, 67, 88, 12, 9, 21, 72, 3, 55, 9, 5, 99, 52, 26, 20, 70, 37, 18, 34, 35, 8, 90, 12, 30, 15, 2, 3, 87, 75, 9, 92, 68, 8, 96, 81, 54, 42, 18, 13, 52, 12, 56, 38, 82, 61, 10, 11, 40, 92, 14, 18, 63, 28, 22, 24, 46, 20, 52, 31, 77, 47, 54, 45, 18, 59, 55, 19, 86, 39, 93, 2, 14, 2, 47, 69, 93, 26, 5, 43, 17, 38, 90, 39, 0, 35, 57, 34, 61, 17, 89, 35, 23, 40, 82, 70, 9, 49, 15, 53, 74, 23, 38, 2, 72, 81, 28, 79, 75, 41, 29, 34, 4, 97, 68, 69, 22, 18, 90, 45, 91, 23, 61, 33, 28, 27, 67, 46, 29, 72, 8, 72, 77, 10, 21, 24, 40, 87, 13, 56, 30, 61, 73, 90, 74, 39, 62, 80, 3, 92, 65, 47, 55, 60, 80, 47, 57, 11, 92, 6, 5, 39, 35, 5, 77, 16, 30, 35, 10, 2, 81, 19, 83, 35, 17, 69, 10, 80, 22, 84, 0, 33, 42, 50, 18, 70, 58, 63, 97, 20, 29, 2, 74, 35, 26, 10, 78, 14, 65, 56, 83, 22, 7, 98, 48, 6, 53, 83, 98, 42, 31, 92, 0, 81, 34, 51, 47, 69, 40, 77, 93, 46, 94, 38, 75, 26, 82, 19, 46, 7, 78, 59, 99, 81, 37, 51, 69, 78, 57, 66, 2, 99, 0, 89, 9, 37, 63, 98, 87, 48, 32, 87, 13, 10, 87, 73, 93, 87, 63, 70, 98, 92, 69, 36, 37, 33, 46, 11, 6, 56, 53, 69, 59, 30, 79, 95, 59, 60, 38, 94, 90, 47, 11, 92, 16, 59, 56, 21, 19, 6, 2, 65, 81, 27, 19, 34, 54, 69, 42, 78, 81, 62, 81, 95, 13, 96, 43, 34, 43, 6, 47, 61, 79, 50, 86, 59, 19, 34, 1, 44, 82, 48, 89, 46, 28, 66, 26, 51, 9, 98, 65, 92, 41, 58, 97, 37, 47, 36, 72, 30, 5, 28, 99, 41, 54, 86, 27, 97, 29, 73, 34, 27, 22, 92, 21, 91, 66, 93, 3, 5, 7, 3, 87, 92, 20, 60, 5, 39, 22, 66, 74, 78, 16, 37, 12, 31, 89, 94, 8, 14, 43, 75, 51, 42, 50, 90, 23, 13, 77, 14, 47, 1, 24, 12, 80, 55, 10, 79, 61, 33, 69, 78, 93, 79, 73, 41, 24, 43, 93, 58, 99, 22, 84, 33, 96, 44, 18, 79, 54, 63, 1, 35, 0, 99, 43, 1, 48, 29, 97, 91, 66, 34, 61, 13, 33, 69, 91, 41, 85, 81, 25, 43, 38, 36, 29, 1, 66, 47, 55, 2, 1, 95, 3, 92, 24, 43, 20, 19, 59, 98, 86, 64, 70, 79, 78, 22, 39, 84, 83, 36, 33, 65, 11, 56, 92, 9, 50, 30, 99, 45, 61, 15, 90, 99, 71, 7, 94, 96, 24, 65, 49, 82, 12, 63, 74, 29, 75, 34, 45, 36, 48, 14, 51, 2, 19, 81, 97, 62, 38, 64, 57, 23, 75, 87, 76, 0, 16, 6, 1, 43, 91, 79, 87, 5, 60, 11, 31, 22, 80, 87, 92, 32, 43, 32, 3, 21, 26, 50, 84, 52, 77, 44, 46, 17, 82, 29, 79, 30, 75, 72, 19, 88, 68, 50, 3, 90, 13, 64, 67, 90, 63, 35, 49, 22, 16, 1, 36, 2, 47, 72, 87, 42, 44, 1, 88, 51, 70, 33, 51, 89, 39, 50, 71, 23, 76, 5, 62, 2, 73, 35, 68, 51, 97, 6, 18, 92, 38, 33, 52, 45, 76, 98, 73, 63, 71, 31, 89, 32, 34, 99, 12, 76, 29, 47, 29, 56, 43, 57, 65, 41, 32, 67, 61, 39, 37, 42, 1, 27, 6, 96, 26, 69, 6, 97, 2, 14, 45, 27, 3, 25, 46, 44, 78, 33, 16, 63, 13, 18, 68, 56, 38, 75, 43, 64, 58, 12, 36, 29, 32, 51, 39, 59, 22, 84, 79, 76, 70, 32, 15, 3, 33, 47, 51, 91, 33, 34, 33, 30, 59, 27, 27, 4, 79, 76, 50, 26, 86, 76, 61, 31, 77, 59, 82, 13, 48, 80, 7, 55, 42, 61, 17, 10, 99, 64, 33, 77, 93, 92, 4, 5, 51, 17, 37, 76, 15, 19, 48, 14, 23, 27, 50, 90, 98, 49, 2, 15, 1, 38, 42, 2, 76, 41, 46, 53, 33, 64, 51, 16, 38, 23, 40, 39, 25, 45, 22, 61, 38, 49, 30, 83, 28, 15, 83, 60, 97, 51, 24, 48, 87, 58, 11, 62, 97, 20, 54, 70, 87, 93, 6, 2, 17, 42, 46, 81, 30, 87, 45, 94, 75, 82, 43, 50, 73, 43, 13, 72, 69, 38, 63, 88, 63, 24, 82, 54, 33, 13, 82, 24, 25, 41, 70, 71, 40, 7, 81, 74, 64, 38, 43, 83, 84, 2, 56, 26, 18, 62, 0, 71, 6, 14, 60, 75, 69, 75, 96, 14, 73, 42, 98, 76, 95, 93, 58, 74, 84, 8, 66, 18, 76, 16, 63, 53, 20, 97, 1, 51, 99, 81, 79, 44, 72, 22, 11, 78, 62, 4, 69, 87, 39, 61, 93, 76, 61, 49, 58, 3, 30, 38, 40, 81, 95, 19, 90, 2}; cout << reduceSum(nums);	6464610	6464610	✔

	Test	Expected	Got	
✔	vector<int> nums {75, 90, 93, 25, 41, 64, 59, 23, 59, 76, 71, 60, 12, 24, 92, 96, 89, 63, 80, 11, 3, 97, 31, 30, 16, 29, 72, 56, 99, 72, 83, 13, 69, 43, 44, 13, 10, 55, 87, 88, 91, 90, 17, 9, 5, 65, 26, 41, 41, 43, 30, 86, 82, 77, 49, 39, 20, 48, 86, 62, 18, 15, 43, 7, 54, 86, 19, 38, 27, 57, 56, 60, 28, 17, 84, 46, 11, 11, 42, 3, 85, 67, 40, 43, 8, 48, 19, 91, 81, 65, 25, 51, 26, 43, 46, 54, 87, 52, 46, 52, 61, 55, 94, 38, 58, 45, 1, 50, 48, 68, 53, 73, 4, 37, 30, 45, 6, 28, 94, 94, 38, 54, 65, 99, 85, 70, 71, 33, 82, 12, 36, 4, 62, 58, 72, 71, 6, 66, 47, 97, 83, 26, 42, 19, 82, 58, 75, 56, 53, 47, 65, 30, 65, 24, 0, 79, 2, 98, 43, 68, 24, 40, 30, 33, 96, 92, 70, 13, 22, 75, 67, 56, 31, 30, 70, 90, 70, 70, 44, 65, 71, 25, 29, 97, 97, 23, 47, 59, 33, 80, 89, 64, 65, 10, 24, 11, 77, 69, 5, 66, 63, 69, 57, 89, 90, 44, 37, 8, 18, 53, 28, 55, 34, 36, 93, 2, 70, 99, 53, 33, 18, 24, 46, 68, 18, 36, 5, 35, 71, 23, 37, 78, 82, 17, 72, 27, 75, 52, 50, 80, 73, 55, 75, 38, 97, 7, 38, 62, 88, 71, 48, 91, 44, 90, 61, 4, 97, 68, 81, 32, 56, 40, 1, 39, 93, 88, 57, 87, 14, 90, 89, 19, 15, 23, 59, 23, 31, 49, 16, 76, 51, 97, 60, 84, 85, 57, 83, 13, 16, 22, 89, 22, 62, 26, 2, 51, 18, 25, 51, 19, 87, 13, 44, 19, 24, 14, 42, 8, 22, 95, 22, 84, 59, 99, 2, 61, 56, 59, 4, 74, 54, 24, 21, 22, 32, 7, 24, 30, 36, 3, 79, 34, 86, 34, 8, 83, 45, 84, 12, 60, 5, 65, 25, 74, 76, 42, 14, 92, 30, 50, 59, 15, 86, 96, 92, 75, 26, 59, 54, 66, 62, 37, 92, 37, 55, 81, 88, 95, 88, 93, 46, 68, 32, 95, 0, 43, 3, 90, 59, 84, 21, 35, 84, 41, 26, 36, 43, 53, 88, 84, 88, 97, 46, 76, 67, 26, 87, 70, 26, 16, 62, 73, 48, 86, 3, 12, 96, 2, 66, 10, 89, 98, 62, 5, 97, 35, 81, 74, 37, 24, 6, 33, 63, 6, 99, 40, 67, 31, 20, 37, 38, 17, 75, 87, 9, 61, 98, 62, 72, 3, 62, 76, 18, 60, 37, 10, 51, 57, 59, 43, 95, 49, 56, 77, 23, 18, 33, 85, 17, 8, 79, 5, 17, 15, 4, 41, 77, 2, 85, 75, 89, 79, 88, 26, 12, 25, 43, 44, 41, 15, 0, 50, 29, 51, 26, 73, 19, 40, 92, 39, 97, 8, 37, 48, 95, 72, 19, 73, 30, 79, 8, 68, 23, 9, 2, 41, 31, 5, 26, 29, 69, 26, 76, 69, 42, 15, 20, 33, 44, 84, 52, 61, 10, 43, 80, 28, 79, 62, 94, 24, 61, 26, 31, 13, 81, 45, 30, 65, 56, 81, 16, 3, 79, 85, 59, 42, 26, 20, 27, 21, 61, 14, 89, 53, 53, 53, 46, 94, 14, 37, 31, 41, 29, 35, 99, 33, 19, 24, 30, 17, 30, 88, 55, 22, 88, 33, 53, 18, 98, 6, 81, 20, 79, 3, 11, 49, 9, 6, 85, 13, 55, 91, 68, 74, 44, 93, 70, 74, 30, 21, 60, 19, 14, 58, 73, 32, 88, 89, 94, 13, 66, 9, 67, 3, 78, 34, 90, 53, 55, 35, 67, 75, 63, 66, 8, 5, 42, 52, 89, 98, 82, 60, 43, 57, 0, 20, 76, 99, 74, 49, 50, 15, 61, 94, 86, 91, 77, 36, 66, 43, 59, 32, 43, 86, 41, 13, 83, 59, 84, 88, 81, 22, 14, 10, 98, 28, 95, 97, 58, 5, 23, 53, 90, 11, 40, 52, 20, 30, 40, 47, 44, 62, 75, 11, 76, 8, 30, 33, 28, 92, 54, 74, 52, 50, 43, 50, 83, 92, 40, 37, 58, 88, 10, 90, 44, 31, 15, 3, 3, 12, 68, 21, 26, 25, 54, 26, 16, 19, 82, 97, 9, 32, 27, 45, 92, 70, 25, 36, 49, 85, 90, 88, 32, 73, 92, 23, 45, 69, 89, 28, 13, 59, 37, 22, 82, 49, 34, 16, 94, 55, 52, 61, 57, 11, 68, 29, 40, 93, 55, 36, 80, 7, 66, 55, 84, 67, 60, 28, 58, 90, 22, 37, 80, 69, 98, 89, 46, 22, 44, 54, 54, 2, 73, 89, 36, 28, 0, 82, 83, 16, 5, 67, 96, 97, 76, 73, 70, 4, 19, 26, 78, 98, 99, 1, 51, 68, 8, 41, 52, 68, 3, 71, 60, 34, 48, 58, 74, 88, 36, 25, 91, 52, 52, 52, 45, 0, 15, 39, 58, 66, 8, 48, 68, 64, 64, 53, 86, 62, 67, 77, 66, 14, 56, 27, 88, 45, 58, 55, 68, 99, 47, 70, 15, 39, 48, 60, 44, 24, 75, 44, 68, 54, 13, 38, 90, 66, 80, 19, 61, 47, 10, 38, 54, 58, 70, 78, 30, 45, 22, 13, 76, 69, 94, 91, 91, 97, 52, 74, 39, 21, 1, 72, 32, 60, 19, 85, 35, 78, 40, 22, 36, 73, 17, 46, 79, 30, 70, 41, 31, 3, 33, 43, 91, 14, 29, 58, 35, 36, 97, 86, 43, 6, 97, 39, 66, 92, 28, 88, 30, 17, 87, 79, 51, 75, 77, 32, 37, 71, 72, 59, 45, 2, 26, 33, 73, 13, 27, 34, 11, 82, 23, 0, 4, 37, 58, 54, 39, 46, 44, 59, 33, 3, 30, 1, 62, 40, 50, 73, 56, 17, 15, 73, 13, 16, 11, 51, 47, 23, 89, 80, 3, 20, 44, 33, 67, 4, 1, 20, 61, 37, 10, 54, 67, 20, 60, 28, 9, 77, 3, 56, 63, 11, 95, 49, 92, 3, 65, 42, 99, 65, 38, 82, 39, 14, 83, 44 ...snip... 76, 15, 31, 97, 84, 22, 40, 15, 50, 3, 87, 22, 72, 90, 21, 15, 16, 23, 21, 86, 37, 44, 94, 84, 95, 19, 61, 59, 82, 7, 48, 12, 9, 41, 47, 78, 39, 82, 34, 13, 6, 72, 88, 33, 74, 56, 70, 12, 96, 49, 79, 57, 59, 10, 94, 62, 3, 24, 13, 5, 93, 52, 89, 26, 22, 8, 97, 96, 39, 84, 56, 45, 95, 83, 89, 95, 67, 57, 5, 21, 67, 43, 99, 37, 18, 97, 58, 3, 76, 36, 85, 33, 6, 88, 55, 28, 10, 87, 86, 64, 3, 19, 80, 80, 83, 72, 2, 32, 67, 30, 37, 98, 87, 98, 94, 54, 54, 46, 90, 77, 92, 94, 13, 37, 42, 16, 38, 35, 22, 95, 60, 24, 25, 8, 12, 28, 16, 89, 83, 36, 30, 70, 38, 51, 94, 49, 82, 4, 11, 53, 25, 42, 5, 82, 74, 39, 90, 59, 59, 9, 58, 20, 49, 22, 63, 58, 52, 86, 32, 16, 65, 27, 11, 31, 76, 32, 17, 5, 34, 79, 0, 73, 59, 68, 96, 90, 91, 61, 39, 98, 33, 27, 74, 89, 25, 32, 49, 2, 54, 3, 17, 13, 95, 29, 7, 86, 16, 82, 3, 81, 55, 71, 71, 83, 48, 8, 61, 47, 0, 57, 24, 72, 5, 53, 10, 38, 2, 64, 85, 32, 12, 38, 11, 17, 30, 94, 47, 5, 13, 83, 54, 2, 39, 4, 42, 86, 13, 99, 88, 92, 46, 81, 72, 78, 77, 32, 25, 13, 10, 56, 74, 59, 6, 66, 80, 32, 5, 74, 64, 12, 87, 4, 56, 95, 85, 2, 2, 75, 64, 92, 13, 66, 88, 52, 54, 0, 83, 86, 68, 95, 56, 50, 55, 23, 46, 49, 34, 41, 32, 58, 29, 86, 19, 30, 50, 33, 5, 53, 74, 33, 80, 18, 59, 92, 26, 85, 7, 81, 19, 61, 71, 11, 25, 17, 5, 53, 6, 94, 11, 1, 71, 92, 17, 68, 16, 59, 96, 28, 88, 46, 56, 33, 62, 63, 71, 28, 76, 34, 39, 57, 39, 43, 22, 59, 76, 93, 36, 1, 85, 24, 29, 71, 91, 61, 67, 21, 82, 41, 6, 31, 37, 67, 10, 75, 40, 58, 49, 38, 63, 46, 30, 85, 61, 42, 61, 43, 53, 82, 0, 11, 80, 45, 97, 60, 39, 46, 60, 98, 50, 6, 48, 3, 33, 2, 73, 46, 9, 15, 73, 0, 22, 37, 91, 81, 33, 44, 11, 12, 41, 20, 80, 81, 14, 80, 6, 7, 7, 29, 80, 43, 43, 0, 73, 43, 30, 69, 13, 58, 22, 30, 82, 91, 38, 51, 76, 85, 57, 23, 22, 86, 2, 7, 1, 38, 3, 65, 6, 33, 36, 56, 11, 23, 85, 18, 54, 92, 34, 86, 6, 49, 94, 83, 98, 69, 34, 4, 3, 91, 71, 62, 24, 24, 79, 75, 88, 64, 95, 20, 27, 98, 37, 5, 77, 31, 57, 72, 83, 65, 24, 99, 96, 12, 37, 66, 99, 62, 82, 72, 97, 90, 38, 97, 70, 30, 25, 24, 35, 63, 99, 45, 90, 65, 6, 22, 33, 82, 69, 22, 40, 83, 61, 94, 94, 66, 87, 91, 46, 74, 66, 0, 18, 24, 51, 48, 32, 14, 38, 86, 6, 33, 82, 61, 3, 74, 80, 82, 44, 17, 96, 8, 57, 78, 80, 50, 17, 17, 45, 82, 91, 80, 79, 30, 30, 61, 93, 79, 1, 21, 93, 90, 7, 59, 9, 98, 90, 74, 9, 29, 74, 99, 67, 90, 17, 17, 29, 11, 21, 65, 74, 80, 66, 32, 60, 71, 64, 62, 12, 6, 22, 46, 46, 98, 72, 88, 38, 5, 63, 21, 56, 34, 91, 17, 48, 72, 94, 45, 5, 24, 94, 67, 95, 7, 30, 55, 25, 64, 25, 43, 52, 61, 28, 51, 62, 71, 11, 85, 23, 14, 21, 7, 26, 75, 67, 14, 26, 14, 10, 13, 75, 67, 28, 71, 26, 36, 56, 49, 71, 47, 17, 43, 50, 24, 69, 8, 52, 38, 66, 13, 48, 38, 17, 2, 69, 55, 74, 27, 54, 16, 78, 14, 91, 83, 86, 44, 13, 56, 9, 85, 37, 94, 52, 95, 37, 91, 81, 62, 36, 89, 54, 18, 52, 39, 55, 17, 80, 46, 23, 84, 42, 2, 61, 79, 45, 13, 1, 51, 96, 1, 67, 76, 75, 7, 6, 87, 42, 6, 58, 33, 83, 18, 90, 8, 42, 95, 27, 2, 36, 53, 63, 32, 68, 73, 77, 81, 14, 31, 73, 43, 45, 25, 66, 45, 46, 92, 63, 61, 15, 30, 43, 48, 36, 80, 28, 80, 17, 65, 45, 13, 19, 78, 42, 80, 32, 79, 96, 35, 77, 78, 10, 61, 22, 63, 28, 85, 93, 84, 6, 92, 10, 5, 63, 40, 19, 98, 58, 90, 38, 92, 72, 84, 27, 54, 16, 27, 96, 57, 43, 94, 98, 58, 52, 8, 86, 30, 27, 31, 0, 14, 92, 38, 39, 25, 39, 84, 1, 62, 41, 81, 27, 2, 94, 20, 36, 17, 72, 94, 22, 88, 26, 22, 86, 11, 10, 73, 90, 28, 56, 28, 32, 20, 77, 16, 36, 60, 39, 29, 24, 91, 73, 26, 68, 2, 13, 0, 29, 0, 72, 60, 62, 71, 9, 18, 56, 74, 34, 4, 49, 36, 31, 24, 32, 65, 25, 33, 39, 36, 86, 31, 83, 5, 86, 80, 7, 12, 53, 77, 74, 1, 10, 30, 30, 17, 76, 59, 73, 13, 6, 40, 71, 86, 52, 42, 84, 49, 31, 90, 44, 23, 58, 53, 11, 56, 83, 32, 82, 40, 60, 14, 43, 18, 87, 32, 36, 28, 94, 64, 43, 66, 15, 10, 61, 90, 71, 14, 20, 0, 65, 42, 5, 11, 81, 31, 90, 28, 70, 85, 96, 70, 29, 77, 61, 45, 39, 81, 51, 11, 20, 54, 20, 31, 17, 1, 17, 76, 95, 14, 53, 14, 29, 2, 76, 67, 40, 78, 43, 5, 21, 8, 46, 74, 43, 74, 23, 48, 1, 22, 76, 29, 30, 96, 36, 42, 7, 22, 34, 88, 85, 51, 34}; cout << reduceSum(nums);	37941342	37941342	✔

	Test	Expected	Got	
✔	vector<int> nums {55, 96, 41, 36, 43, 3, 10, 7, 35, 17, 89, 54, 36, 71, 58, 41, 86, 87, 18, 2, 92, 20, 95, 97, 30, 15, 8, 33, 86, 85, 80, 64, 48, 78, 20, 94, 75, 58, 22, 64, 67, 6, 7, 42, 24, 5, 17, 3, 18, 71, 61, 85, 91, 38, 31, 25, 27, 7, 52, 32, 78, 31, 27, 85, 87, 54, 91, 85, 79, 49, 79, 40, 79, 71, 99, 22, 54, 85, 46, 10, 38, 70, 65, 10, 2, 51, 55, 19, 60, 20, 1, 6, 57, 39, 60, 55, 29, 42, 97, 41, 55, 35, 88, 7, 84, 2, 92, 37, 59, 42, 54, 5, 37, 68, 0, 87, 13, 17, 96, 88, 59, 26, 96, 31, 93, 16, 88, 74, 67, 92, 3, 18, 15, 65, 62, 11, 95, 89, 7, 91, 85, 90, 16, 77, 3, 20, 94, 91, 32, 6, 21, 55, 51, 68, 1, 72, 43, 88, 59, 20, 55, 61, 0, 41, 91, 75, 93, 82, 86, 93, 83, 80, 29, 31, 53, 29, 96, 84, 0, 55, 80, 83, 39, 84, 96, 28, 35, 28, 12, 65, 96, 15, 83, 77, 37, 89, 95, 17, 72, 92, 35, 78, 21, 33, 4, 54, 25, 37, 0, 18, 62, 9, 91, 27, 65, 19, 86, 84, 11, 19, 18, 80, 28, 63, 37, 32, 17, 7, 56, 99, 3, 87, 16, 11, 13, 37, 4, 35, 63, 61, 71, 55, 79, 76, 54, 40, 24, 85, 13, 57, 96, 34, 99, 85, 94, 6, 39, 61, 5, 97, 16, 16, 32, 33, 71, 98, 2, 8, 39, 66, 11, 34, 93, 95, 76, 76, 70, 8, 43, 81, 66, 35, 19, 84, 6, 15, 39, 6, 8, 98, 16, 29, 68, 13, 72, 2, 3, 77, 96, 59, 68, 30, 50, 71, 55, 74, 0, 90, 89, 4, 63, 88, 99, 10, 27, 77, 70, 31, 46, 30, 91, 81, 31, 63, 89, 26, 18, 44, 19, 57, 57, 25, 66, 60, 68, 97, 3, 32, 92, 25, 77, 56, 96, 92, 1, 9, 29, 18, 9, 10, 76, 59, 10, 51, 50, 53, 33, 39, 68, 89, 57, 86, 2, 42, 23, 12, 96, 46, 22, 44, 79, 53, 16, 44, 57, 63, 61, 65, 57, 38, 54, 14, 29, 52, 12, 88, 64, 85, 42, 28, 58, 44, 52, 80, 78, 38, 46, 58, 3, 0, 29, 59, 82, 3, 36, 0, 43, 17, 61, 79, 98, 61, 53, 27, 12, 77, 77, 3, 54, 39, 71, 48, 73, 94, 40, 90, 15, 70, 55, 26, 1, 87, 34, 25, 13, 70, 52, 75, 17, 40, 55, 20, 65, 43, 34, 87, 6, 51, 21, 50, 27, 85, 23, 17, 29, 38, 21, 42, 5, 57, 54, 95, 72, 41, 87, 26, 39, 74, 55, 46, 56, 6, 25, 63, 93, 65, 9, 84, 66, 0, 88, 15, 80, 61, 52, 97, 89, 48, 63, 90, 65, 12, 31, 99, 79, 12, 98, 86, 84, 63, 95, 1, 7, 27, 89, 19, 79, 38, 6, 28, 21, 94, 90, 87, 33, 94, 93, 9, 21, 12, 31, 67, 99, 74, 86, 24, 52, 28, 13, 96, 66, 50, 27, 88, 59, 61, 62, 67, 38, 35, 1, 13, 6, 85, 40, 94, 97, 13, 30, 57, 3, 95, 87, 69, 38, 47, 3, 11, 32, 27, 75, 85, 90, 7, 63, 39, 8, 14, 87, 24, 47, 86, 18, 24, 34, 32, 76, 99, 40, 1, 63, 32, 80, 40, 64, 63, 34, 95, 51, 16, 98, 91, 92, 85, 87, 32, 59, 69, 57, 33, 82, 78, 37, 22, 34, 96, 88, 39, 43, 4, 23, 6, 76, 47, 47, 99, 22, 91, 1, 43, 8, 54, 39, 15, 72, 96, 79, 55, 55, 88, 83, 79, 90, 49, 97, 64, 35, 51, 93, 58, 82, 75, 62, 36, 97, 35, 98, 41, 26, 91, 37, 46, 7, 23, 50, 72, 55, 90, 10, 31, 75, 50, 37, 55, 55, 33, 43, 66, 62, 99, 81, 18, 63, 53, 96, 32, 42, 10, 5, 9, 98, 8, 70, 16, 29, 41, 68, 13, 81, 6, 49, 12, 55, 68, 12, 45, 53, 89, 10, 3, 36, 33, 91, 58, 96, 14, 58, 21, 7, 12, 49, 55, 68, 90, 81, 66, 0, 81, 68, 66, 22, 58, 56, 11, 84, 37, 7, 77, 60, 19, 95, 51, 27, 69, 4, 62, 47, 70, 67, 68, 69, 31, 82, 84, 59, 52, 45, 99, 34, 67, 35, 39, 46, 24, 82, 3, 28, 11, 27, 56, 84, 57, 81, 79, 21, 43, 7, 95, 22, 68, 4, 23, 54, 46, 95, 71, 94, 52, 59, 47, 89, 42, 93, 85, 25, 21, 99, 89, 8, 24, 87, 33, 97, 7, 13, 82, 84, 74, 29, 96, 24, 30, 5, 40, 40, 84, 93, 72, 31, 76, 11, 7, 43, 8, 5, 31, 59, 36, 9, 31, 25, 69, 40, 37, 40, 47, 90, 20, 2, 72, 70, 17, 84, 74, 97, 27, 85, 98, 79, 56, 57, 85, 12, 98, 19, 79, 80, 22, 48, 27, 30, 91, 93, 54, 50, 6, 15, 20, 69, 13, 6, 95, 41, 90, 72, 61, 28, 58, 95, 56, 38, 70, 60, 43, 67, 72, 57, 28, 61, 40, 61, 52, 78, 21, 78, 11, 84, 62, 27, 73, 66, 74, 12, 60, 90, 24, 19, 51, 44, 67, 60, 49, 25, 2, 92, 82, 99, 41, 80, 11, 37, 87, 28, 40, 54, 35, 94, 20, 51, 65, 45, 59, 84, 14, 16, 99, 10, 70, 57, 7, 74, 49, 86, 87, 5, 88, 73, 6, 29, 46, 94, 4, 35, 3, 47, 59, 53, 43, 71, 52, 70, 25, 3, 98, 20, 2, 98, 31, 65, 97, 50, 31, 41, 29, 60, 96, 46, 77, 87, 18, 97, 87, 86, 38, 56, 42, 80, 9, 61, 84, 20, 64, 30, 92, 3, 46, 61, 5, 18, 66, 58, 44, 58, 61, 26, 90, 17, 29, 59, 92, 82, 95, 66, 26, 24, 31, 44, 38, 57, 30, 51, 55, 69, 96, 62, 34, 76, 98, 84, 15, 56 ...snip... 68, 42, 2, 31, 21, 30, 66, 45, 77, 2, 12, 2, 11, 66, 0, 36, 65, 58, 7, 9, 63, 26, 77, 59, 22, 12, 47, 47, 47, 75, 55, 56, 1, 15, 87, 2, 89, 73, 15, 51, 80, 43, 8, 38, 88, 59, 57, 39, 15, 61, 27, 3, 19, 7, 68, 68, 34, 40, 99, 67, 40, 7, 27, 51, 59, 17, 44, 1, 53, 37, 62, 82, 45, 76, 5, 96, 39, 13, 5, 65, 25, 97, 39, 8, 42, 78, 52, 26, 93, 56, 28, 0, 31, 40, 97, 98, 20, 5, 2, 67, 7, 36, 62, 89, 99, 1, 1, 20, 88, 15, 16, 98, 71, 34, 28, 30, 47, 87, 33, 73, 37, 23, 36, 32, 49, 85, 35, 15, 84, 94, 67, 1, 52, 58, 70, 30, 35, 44, 3, 41, 69, 25, 59, 58, 48, 5, 28, 91, 20, 98, 51, 17, 34, 65, 74, 5, 29, 58, 8, 91, 48, 96, 32, 51, 63, 5, 76, 31, 28, 27, 2, 9, 51, 87, 32, 0, 13, 1, 21, 0, 58, 52, 79, 69, 9, 71, 37, 80, 80, 92, 16, 50, 28, 52, 38, 21, 22, 55, 35, 89, 15, 69, 72, 30, 85, 54, 68, 86, 45, 42, 73, 89, 28, 32, 83, 43, 41, 88, 79, 37, 18, 94, 59, 88, 99, 93, 8, 67, 49, 99, 31, 3, 23, 15, 36, 92, 20, 57, 68, 10, 49, 9, 98, 5, 6, 34, 36, 25, 40, 24, 27, 44, 34, 84, 1, 1, 94, 15, 13, 32, 77, 6, 84, 43, 69, 36, 98, 41, 19, 36, 94, 5, 17, 45, 91, 54, 78, 35, 71, 68, 63, 84, 61, 53, 92, 94, 38, 56, 0, 43, 75, 46, 42, 91, 78, 47, 60, 36, 46, 72, 16, 16, 44, 75, 68, 54, 99, 22, 66, 86, 55, 95, 32, 98, 9, 71, 66, 1, 66, 26, 66, 27, 97, 91, 55, 10, 29, 25, 97, 43, 16, 36, 87, 92, 44, 55, 11, 36, 48, 54, 34, 51, 87, 36, 59, 64, 83, 59, 65, 60, 49, 15, 95, 82, 95, 68, 0, 67, 53, 87, 0, 82, 60, 2, 81, 76, 92, 59, 77, 84, 49, 10, 23, 7, 33, 85, 44, 34, 7, 61, 12, 68, 79, 3, 48, 81, 41, 1, 71, 11, 31, 49, 16, 72, 10, 94, 22, 89, 79, 61, 91, 21, 94, 5, 12, 73, 56, 44, 78, 27, 26, 13, 33, 71, 48, 34, 45, 62, 47, 97, 33, 1, 9, 65, 82, 62, 87, 20, 73, 97, 65, 10, 64, 64, 23, 77, 87, 81, 19, 15, 77, 51, 94, 31, 92, 23, 17, 59, 93, 88, 49, 17, 29, 41, 0, 53, 25, 59, 85, 75, 39, 79, 74, 26, 5, 24, 28, 12, 73, 75, 52, 2, 14, 14, 52, 54, 2, 60, 40, 63, 25, 20, 27, 78, 69, 12, 49, 39, 66, 67, 53, 9, 99, 24, 30, 61, 5, 41, 13, 51, 8, 76, 46, 87, 53, 85, 25, 44, 11, 25, 43, 71, 0, 40, 77, 46, 77, 77, 98, 17, 76, 50, 56, 24, 38, 34, 21, 74, 85, 15, 67, 22, 0, 83, 97, 44, 13, 91, 79, 43, 6, 42, 80, 40, 99, 3, 11, 9, 82, 40, 25, 51, 92, 37, 4, 41, 16, 52, 35, 58, 21, 11, 60, 73, 96, 41, 29, 91, 75, 56, 72, 16, 20, 79, 5, 33, 57, 82, 93, 9, 20, 50, 36, 75, 1, 47, 27, 36, 68, 71, 28, 78, 80, 52, 77, 71, 21, 5, 67, 78, 6, 33, 70, 83, 88, 80, 4, 81, 7, 83, 32, 5, 34, 38, 41, 60, 21, 96, 99, 37, 66, 57, 84, 0, 19, 23, 86, 70, 71, 43, 65, 30, 99, 69, 59, 47, 23, 74, 99, 55, 1, 5, 86, 44, 41, 40, 43, 77, 53, 96, 14, 23, 59, 92, 55, 60, 53, 38, 40, 76, 85, 67, 16, 2, 37, 4, 38, 33, 8, 61, 5, 24, 46, 76, 33, 59, 32, 77, 82, 39, 59, 20, 15, 63, 39, 68, 25, 0, 55, 39, 37, 50, 39, 34, 4, 24, 29, 43, 90, 86, 72, 19, 38, 96, 46, 53, 27, 6, 40, 44, 61, 90, 95, 0, 22, 45, 65, 20, 79, 62, 20, 98, 91, 72, 87, 79, 86, 94, 71, 48, 81, 68, 47, 10, 50, 19, 26, 51, 10, 88, 36, 13, 18, 82, 68, 66, 42, 9, 42, 28, 95, 91, 69, 9, 1, 41, 22, 14, 12, 13, 18, 82, 58, 63, 40, 69, 53, 37, 66, 18, 68, 67, 69, 72, 20, 30, 79, 40, 33, 15, 81, 20, 91, 71, 14, 34, 46, 43, 43, 64, 68, 17, 18, 62, 81, 9, 0, 13, 95, 1, 73, 27, 75, 21, 91, 50, 35, 28, 88, 41, 62, 42, 67, 65, 77, 86, 86, 28, 20, 84, 91, 32, 36, 82, 73, 6, 98, 96, 46, 32, 88, 0, 44, 44, 82, 15, 97, 22, 63, 98, 6, 50, 17, 60, 92, 13, 67, 28, 46, 72, 70, 57, 49, 66, 34, 84, 76, 57, 69, 66, 75, 1, 12, 58, 67, 36, 57, 37, 88, 9, 37, 37, 9, 5, 67, 51, 55, 23, 4, 84, 77, 45, 20, 6, 3, 91, 9, 70, 46, 20, 2, 17, 58, 65, 43, 99, 17, 92, 20, 91, 7, 41, 19, 8, 58, 77, 26, 24, 29, 68, 31, 87, 84, 83, 14, 30, 17, 10, 8, 89, 25, 61, 58, 62, 4, 22, 89, 23, 98, 66, 35, 69, 3, 29, 74, 49, 41, 63, 31, 71, 61, 30, 26, 34, 98, 22, 8, 15, 28, 39, 45, 29, 9, 42, 75, 72, 64, 7, 95, 93, 3, 37, 86, 14, 67, 58, 99, 30, 69, 89, 94, 3, 8, 4, 96, 90, 40, 23, 42, 99, 48, 38, 99, 46, 91, 87, 78, 37, 63, 40, 99, 14, 80, 50, 5, 27, 64, 30, 52, 25, 11, 42, 16, 54, 29, 63, 13, 30, 89, 41, 8, 39, 24, 44, 78, 32, 83, 79, 40, 31, 1, 82}; cout << reduceSum(nums);	80709014	80709014	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 9

Chính xác

Điểm 1,00 của 1,00

Given an array of non-negative integers. Each time, we can take the smallest integer out of the array, multiply it by 2, and push it back to the array.

Request: Implement function:

```
int leastAfter(vector<int>& nums, int k);
```

Where `nums` is the given array (the length of the array is between 1 and 100000). This function returns the smallest integer in the array after performing the operation `k` times (`k` is between 1 and 100000).

Example:

Given `nums = [2, 3, 5, 7]`.

In the 1st operation, we take `2` out and push back `4`. The array is now `nums = [3, 4, 5, 7]`.

In the 2nd operation, we take `3` out and push back `6`. The array is now `nums = [4, 5, 6, 7]`.

In the 3rd operation, we take `4` out and push back `8`. The array is now `nums = [5, 6, 7, 8]`.

With `k = 3`, the result would be `5`.

Note:

In this exercise, the libraries `iostream, string, cstring, climits, utility, vector, list, stack, queue, map, unordered_map, set, unordered_set, functional, algorithm` has been included and `namespace std` are used. You can write helper functions and classes. Importing other libraries is allowed, but not encouraged, and may result in unexpected errors.

For example:

Test	Result

`vector<int> nums {2, 3, 5, 7};`
`int k = 3;`
`cout << leastAfter(nums, k);`
	5

Answer: (penalty regime: 0, 0, 0, 5, 10, ... %)

Reset answer

```
struct mycmp{
    bool operator()(int a,int b){
        return a>b;
    }
};
int leastAfter(vector<int>& nums, int k) {
    // STUDENT ANSWER
    priority_queue<int,vector<int>,mycmp> pq;
    for(unsigned int i=0;i<nums.size();i++){
        pq.push(nums[i]);
    }
    for(int i=0;i<k;i++){
        int temp=pq.top();
        pq.pop();
        pq.push(temp*2);
    }
    return pq.top();
}
```

	Test	Expected	Got	

✔	`vector<int> nums {2, 3, 5, 7};`
	`int k = 3;`
	`cout << leastAfter(nums, k);`
		5	5	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 10

Chính xác

Điểm 0,20 của 1,00

Given a sequence of sorted integers whose prime factors only include `2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31` and starts at `1`.

Request: Implement function:

```
int uglyNumberIndex(int n);
```

This function returns the `n`-th indexed element of the sequence. (The sequence is `0`-th indexed, and `n` is in range `[0, 100000]`).

Example:

The first elements of the sequence: `[1, 2, 3, 4, 5, 6, 7,...]`

The `2`-nd indexed element is `3`.

Note:

In this exercise, the libraries `iostream, string, cstring, climits, utility, vector, list, stack, queue, map, unordered_map, set, unordered_set, functional, algorithm` has been included and `namespace std` are used. You can write helper functions and class. Importing other libraries is allowed, but not encouraged.

For example:

Test	Result
`int n = 5;` `cout << uglyNumberIndex(n);`	6

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
int uglyNumberIndex(int n) {
    // STUDENT ANSWER
    priority_queue<int,vector<int>,greater<int>> pq;
    int prime[]={2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31 };
    for(int i=0;i<11;i++){
        pq.push(prime[i]);
    }
    int temp=1;
    for(int i=0;i<n;i++){
        temp=pq.top();
        //cout<<temp<<endl;
        pq.pop();
        for(unsigned int j=0;j<sizeof(prime)/sizeof(int);j++){
            pq.push(prime[j]*temp);
        }
        int top=pq.top();
        pq.pop();
        while(top==pq.top()){
            pq.pop();
        }
        pq.push(top);
    }
    return temp;
}
```

	Test	Expected	Got	
✔	`int n = 5;` `cout << uglyNumberIndex(n);`	6	6	✔
✔	`int n = 10;` `cout << uglyNumberIndex(n);`	11	11	✔

	Test	Expected	Got	
✔	`int n = 50;` `cout << uglyNumberIndex(n);`	56	56	✔
✔	`int n = 100;` `cout << uglyNumberIndex(n);`	132	132	✔
✔	`int n = 500;` `cout << uglyNumberIndex(n);`	1242	1242	✔
✔	`int n = 1000;` `cout << uglyNumberIndex(n);`	3757	3757	✔
✔	`int n = 5000;` `cout << uglyNumberIndex(n);`	68121	68121	✔
✔	`int n = 10000;` `cout << uglyNumberIndex(n);`	277704	277704	✔
✔	`int n = 50000;` `cout << uglyNumberIndex(n);`	10848981	10848981	✔
✔	`int n = 100000;` `cout << uglyNumberIndex(n);`	63536000	63536000	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00. Tính toán cho lần làm bài trước đó, điểm **0,20/1,00**.