

**PROJECT**

**Resolving Overbooked Flight**

Department : Computer Science and Engineering

Semester : Fall 2018

Course : CSE207

Course Title : Data Structures

Section : 01

Group members : Mahruf Zaman Utso (2017-1-60-044)

Kazi Imran Ahmed (2017-1-60-110)

Course Instructor : Dr. Maheen Islam

Date of Submission : 10-12-2018

Context

* Introduction
* Algorithm
* Source Code
* Output Screens
* Conclusions

**Introduction**

* An airline company uses the formula to determine the priority of passengers on the waiting list for overbooked flights.
* Priority Number = A/1000+B-C.
* A is the customer’s total mileage in the past year.
* B is the number of years in his or her frequent flier program.
* C is a sequence number representing the customer’s arrival position when he or she booked the flight.
* We were given a data of overbooked customers mentioning their total mileage in last year , number of years in their flier program and sequence number representing their arrival position when they booked flight.
* We have to build an application for the airline company that builds a priority queue using the priority number and prints a list of waiting customers in priority sequence.
* For constructing the application we need to do the following:

a. We need to build a priority queue (binary heap) based on the priority number.

b. We need to build a stack store information of customer who will get flight

ticket later.

In the project we were told to do the following:

* Builds a priority queue using the priority number
* Prints a list of waiting customers in priority sequence.
* Print details of customer who took flight ticket later

**Algorithm: Priority Queue (Binary Heap)**

* We declared a structure name node which contains a strong variable called name and four integer variables called mileage, year, sequence and priority. We calculated Priority Number using this formula,

Priority = (mileage/1000) + year-sequence

After that we sorted the structure using heap and printed it

**Algorithm: Stack (storing the information of customer getting flight ticket later)**

* Firstly We pushed all the information of passengers in the stack according to ascending priority sequence.
* Secondly We took input how many passengers has canceled their ticket
* Finally We print the details of the passenger who will get the flight later by popping data from the stack

**Source Code**

#include<bits/stdc++.h>

#define STACK\_MAX 20

using namespace std;

bool a=1;

struct Stack

{

string data[STACK\_MAX];

int mileage1[STACK\_MAX];

int year1[STACK\_MAX];

int sequence1[STACK\_MAX];

int priority1[STACK\_MAX];

int top;

};

struct node

{

string name;

int mileage;

int year;

int sequence;

int priority;

};

void heapify(struct node arr[], int n, int i)

{

int smallest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[l].priority < arr[smallest].priority)

smallest = l;

if (r < n && arr[r].priority < arr[smallest].priority)

smallest = r;

if (smallest != i) {

swap(arr[i], arr[smallest]);

heapify(arr, n, smallest);

}

}

//Descending order

void heapsort(struct node arr[], int n)

{

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

for (int i = n - 1; i >= 0; i--) {

swap(arr[0], arr[i]);

heapify(arr, i, 0);

}

}

void heapify1(struct node arr1[], int n, int i)

{

int largest = i;

int l = 2\*i + 1;

int r = 2\*i + 2;

if (l < n && arr1[l].priority > arr1[largest].priority)

largest = l;

if (r < n && arr1[r].priority > arr1[largest].priority)

largest = r;

if (largest != i)

{

swap(arr1[i], arr1[largest]);

heapify1(arr1, n, largest);

}

}

//Ascending prder

void heapsort1(struct node arr1[], int n)

{

for (int i = n / 2 - 1; i >= 0; i--)

heapify1(arr1, n, i);

for (int i=n-1; i>=0; i--)

{

swap(arr1[0], arr1[i]);

heapify1(arr1, i, 0);

}

}

void push(Stack \*s,struct node item)

{

if(s->top <STACK\_MAX)

{

s->data[s->top]= item.name;

s->mileage1[s->top]= item.mileage;

s->year1[s->top]= item.year;

s->sequence1[s->top]= item.sequence;

s->priority1[s->top]= item.priority;

s->top= s->top+1;

}

else

{

cout<<"The Stack is full "<<endl;

}

}

void pop(Stack \*s)

{

struct node item;

if(s->top ==0)

{

cout<<"The stack is empty! "<<endl;

a=0;

}

else

{

s->top= s->top-1;

cout<<s->data[s->top]<<"\t"<<s->mileage1[s->top]<<"\t"<<s->year1[s->top]<<"\t"<<s->sequence1[s->top]<<"\t"<<s->priority1[s->top]<<endl;

}

}

int main()

{

struct Stack my\_stack;

my\_stack.top=0;

struct node temp;

struct node \*x,\*y;

int cancel;

int n;

cout<<"Input the number of passengers on the waiting list "<<endl;

cin>>n;

x= new node[n];

y= new node[n];

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

{

cin>>x[i].name>>x[i].mileage>>x[i].year>>x[i].sequence;

x[i].priority= (x[i].mileage/1000)+x[i].year- x[i].sequence; // priority= (mileage/1000)+ years- sequence

}

heapsort(x,n);

cout<<endl<<endl;

cout<<"Name"<<"\t"<<"Mileage"<<"\t"<<"Year"<<"\t"<<"Sequence"<<"\t"<<"Priority"<<"\t"<<endl;

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

{

cout<<x[i].name<<"\t"<<x[i].mileage<<"\t"<<x[i].year<<"\t"<<x[i].sequence<<"\t"<<x[i].priority<<endl;

}

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

{

y[i]= x[i];

}

cout<<endl<<endl;

heapsort1(y,n);

cout<<"Input the number of passengers canceled the flight "<<endl<<endl;

cin>>cancel;

cout<<endl<<endl;

cout<<"Name"<<"\t"<<"Mileage"<<"\t"<<"Year"<<"\t"<<"Sequence"<<"\t"<<"Priority"<<"\t"<<endl;

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

{

temp= y[i];

push(&my\_stack,temp);

}

my\_stack.top=(my\_stack.top)-cancel;

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

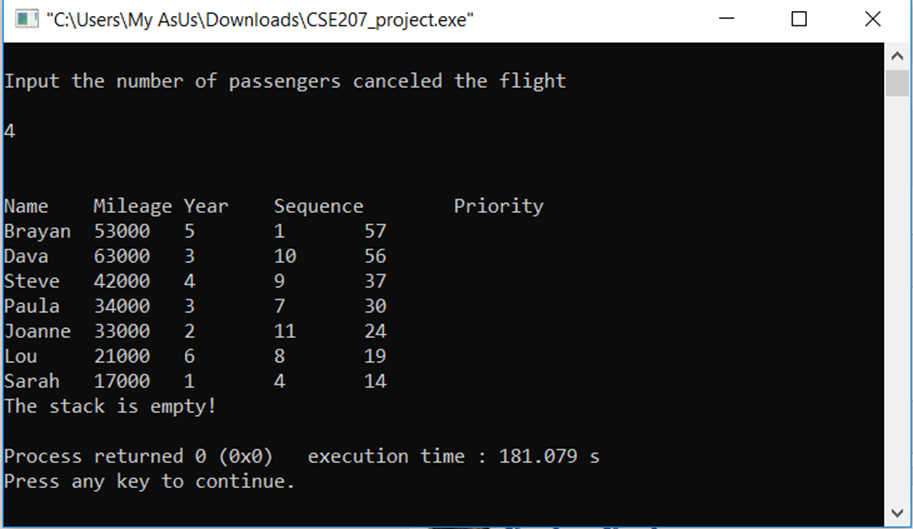
{

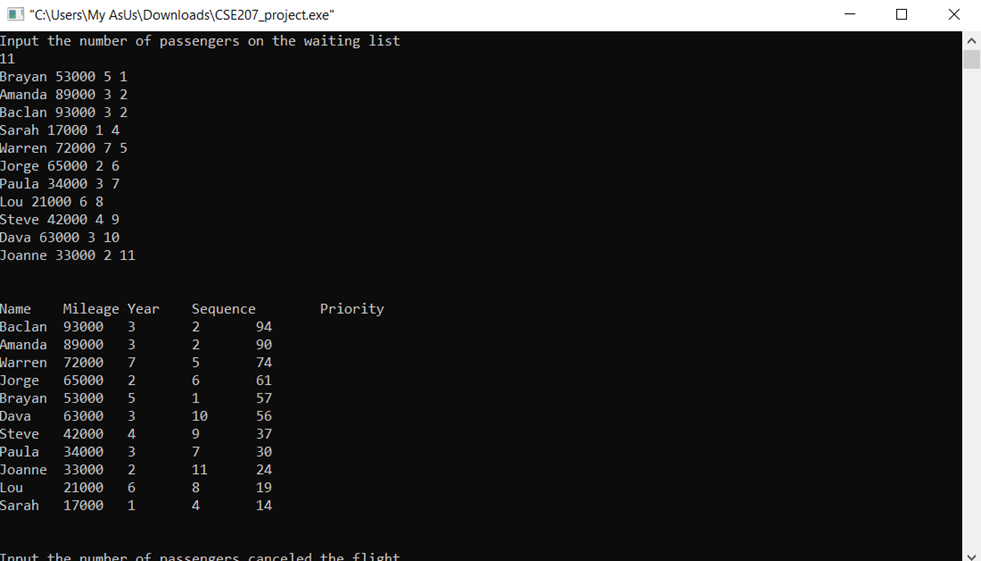
if(a) pop(&my\_stack);

}

}

**Output Screens**





**Drawbacks**

* We faced some problems while doing our projects.
* When we tried to input passengers name in the program application it returned garbage value after entering space between names. That is why we could only able to enter the passengers first name only.
* There is always room for improvement. Once we are finished with the input of data and execute the program ,we can not add more data later in the same program. We can improve this side of the application.
* Despite of these limitations we are able to build an application which could build an priority queue based on the priority number and build a stack to store of customer who will get flight ticket later