Name of Student :- Pavan Ratan Gole

UID:- 2022601003

Batch:- C4

Exp. No. 2

Aim:-

We will reconsider the railroad car rearrangement problem of Section 8.5.3. This time the holding tracks lie between the input and output track as in Figure 9.11. These tracks operate in a FIFO manner and so may be regarded as queues. As in the case of Section 8.5.3, moving a car from a holding track to the input track or from the output track to a holding track is forbidden. All car motion is in the direction indicated by the arrowheads of Figure 9.11.

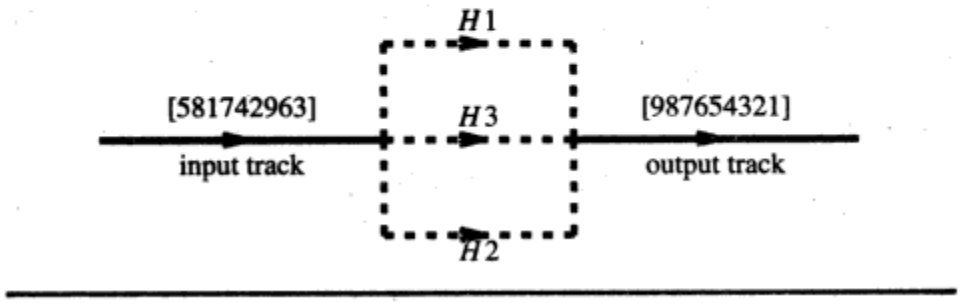

Figure 9.11 A three-track example

Program:-

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

class Queue
{

public:
   int front = -1;
   int rear = -1;
```

```
int *arr;
int size = 4;

Queue(int kize)
{
    arr = new int[kize];
    size = kize;
}

bool isEmpty()
{
    if ((front == -1 && rear == -1))
    {
        return true;
    }
    return false;
}

bool isFull()
{
    if ((size - 1 == rear && front == 0) || front - 1 == rear)
    {
        return true;
    }
    return false;
}

void enqueue(int data)
{
    if (!isFull())
    {
        if (front == -1)
        {
            front = 0;
        }
        if (rear == size - 1)
        {
            rear = -1;
        }
        ++rear;
```

```
            arr[rear] = data;
        }
        else
        {
            cout << "Queue is Full" << endl;
        }
    }

    void dequeue()
    {
        if (!isEmpty())
        {

            if (rear == front)
            {
                arr[front] = -1;
                rear = front = -1;
            }
            else
            {
                arr[front] = -1;
                front++;
                if (front > size - 1)
                {
                    front = 0;
                }
            }
        }
        else
        {
            cout << "Queue is Empty" << endl;
        }
    }

    int peek()
    {
        if (!isEmpty())
        {
            return arr[front];
        }
```

```
        return -1;
    }

    int getlast()
    {
        if (!isEmpty())
        {
            return arr[rear];
        }

        return -1;
    }
};

void solve()
{
    Queue *input = new Queue(9);
    // int inputan[9] = {1,2,3,4,5,6,7,8,9};
    int inputan[9] = {3,6,9,2,4,7,1,8,5};
    //int inputan[9] = {9,8,7,6,5,4,3,2,1};
    int n = sizeof(inputan) / sizeof(inputan[0]);

    int arr[9];
    for (int i = 0; i < 9; i++)
    {
        arr[i] = inputan[i];
    }

    sort(inputan, inputan + n);
    for (int i = 0; i < 9; i++)
    {
        input->enqueue(arr[i]);
    }

    Queue *output = new Queue(9);
    Queue *h1 = new Queue(9);
    Queue *h2 = new Queue(9);
    Queue *h3 = new Queue(9);
    int counter = 0;
```

```
bool issolved = true;
while (!input->isEmpty())
{
    if (input->peek() == inputan[counter])
    {
        output->enqueue(inputan[counter]);
        input->dequeue();
        counter++;
    }


    else if (h1->isEmpty())
    {
        h1->enqueue(input->peek());
        input->dequeue();
    }

    else if (h1->getlast() <= input->peek())
    {
        h1->enqueue(input->peek());
        input->dequeue();
    }
    else if (h2->isEmpty())
    {
        h2->enqueue(input->peek());
        input->dequeue();
    }
    else if (!h2->isEmpty() && h2->getlast() <= input->peek())
    {
        h2->enqueue(input->peek());
        input->dequeue();
    }
    else if (h3->isEmpty())
    {
        h3->enqueue(input->peek());
        input->dequeue();
    }
    else if (!h3->isEmpty() && h3->getlast() <= input->peek())
    {
        h3->enqueue(input->peek());
```

```
            input->dequeue();
        }
        else
        {
            cout << "Problem cannot be solved" << endl;
            issolved = false;
            break;
        }
    }

    if (issolved)
    {
        while (!h1->isEmpty() || !h2->isEmpty() || !h3->isEmpty())
        {
            if (h1->peek() == inputan[counter])
            {
                output->enqueue(inputan[counter]);
                h1->dequeue();
                counter++;
            }

            else if (h2->peek() == inputan[counter])
            {
                output->enqueue(inputan[counter]);
                h2->dequeue();
                counter++;
            }

            else if (h3->peek() == inputan[counter])
            {
                output->enqueue(inputan[counter]);
                h3->dequeue();
                counter++;
            }
        }
    }

    while (!output->isEmpty() && issolved)
    {
        cout << output->peek() << " ";
```

```
        output->dequeue();
    }
    cout << endl;
}

int main(int argc, char const *argv[])
{
    solve();
}
```

Output:-

1)

```
9,8,7,6,5,4,3,2,1
```

```
Problem cannot be solved
```

2)

```
3, 6, 9, 2, 4, 7, 1, 8, 5
```

```
1 2 3 4 5 6 7 8 9
```