Lab5 Solutions

Note: ในเฉลยของแลปนี้จะใช้ Queue และ Priority queue ของ Standard Template Library(STL) แทนการ implement เองโดยใช้ linked list จะถือว่าเข้าใจหลักการเขียน queue มาบ้างแล้วนิดนึง อีกอย่างโค้ดตัวอย่างจะได้ไม่ยาวเกิน มีคำสั่งที่ใช้บ่อยประมาณนี้

- queue<variable type> name — ประกาศ stack ชื่อ name
- name.push(variable) — เพิ่มข้อมูล
- name.front() | name.top() — ดูข้อมูลตัวแรก (ของ queue | priority queue)
- name.pop() — ลบข้อมูลที่อยู่ด้านหน้าสุดออก

ศึกษาเพิ่มเติมได้ที่ https://cplusplus.com/reference/queue/

Dynamic Queue

Implement queue เพื่อรับคำสั่งทั้งหมด 4 คำสั่ง ระวังเรื่องการ pop queue ที่ว่างและการเข้าถึงข้อมูลตัวแรกตอนที่ queue ว่างอาจทำให้ error ได้

Example Code

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

int main()
{
    queue<int> queue;
    int n,x,a;
    cin >> n;

    for(int i=0;i<n;i++)
    {
        cin >> x;
        if(x==1)
        {
            cin >> a;
            queue.push(a);
        }
```

```
        else if(x==2 && !queue.empty()) queue.pop();
        else if(x==3 && !queue.empty()) cout << queue.front() << endl;
        else if(x==3 && queue.empty()) cout << "EMPTY" << endl;
        else if(x==4) cout << queue.size() << endl;
    }
}
```

$time\ complexity : O(n)$

ATM

สร้าง struct Person เพื่อเก็บข้อมูลของคนที่มากดเงินและ implement queue ของคนที่มาต่อแถว โดยเริ่มจากการ *pop* คนแรกใน *queue* ออกมา ถ้าหากกดเงิน *x* บาทแล้วพอให้แสดงหมายเลขของคนนั้น ถ้าหากยังไม่พอให้ *push* เข้าไปใน *queue* ใหม่ ทำแบบนี้จนกว่า *queue* จะว่าง

Example Code

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

typedef struct _node{
    int index;
    int x;
} Person;

int main()
{
    queue<Person> queue;
    int n,x;
    cin >> n >> x;
    for(int i=1;i<=n;i++)
    {
        Person temp = Person();
        cin >> temp.x;
        temp.index = i;
        queue.push(temp);
    }

    while(!queue.empty())
    {
        Person temp = queue.front();
        queue.pop();
        temp.x -= x;

        if(temp.x > 0) queue.push(temp);
        else cout << temp.index << " ";
    }
}
```

$time\ complexity : O(\frac{n}{x} \cdot \max(a))$

Ice-cream Shop

ถ้าหากต้องการขายไอศกรีมให้ลูกค้าได้จำนวนมากที่สุด ควรขายให้ลูกค้าที่รอได้เป็นเวลาน้อยที่สุดก่อน ในกรณีนี้สามารถใช้ Priority queue เข้ามาช่วยได้ จากนั้นพิจารณาลูกค้าทีละคนว่าสามารถขายให้ได้หรือไม่ โดยวิธีคือ เวลาที่ลูกค้าคนนั้นรอต้องน้อยกว่าจำนวนไอศกรีมที่ขายไปแล้ว(คิดแบบนี้ได้เพราะว่าการขายไอศกรีมให้ลูกค้า 1 คนใช้เวลา 1 นาทีพอดี) และน้อยกว่าเวลาที่จะปิดร้าน ถึงจะขายให้ได้

Example Code

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

int main()
{
    int n,t,count=0;
    cin >> n >> t;

    priority_queue<int> queue;
    for(int i=0;i<n;i++)
    {
        int p;
        cin >> p;
        queue.push(p);
    }

    for(int i=0;i<n;i++)
    {
        int front = queue.top();
        if(count < min(front,t)) count++;
    }
    cout << count;
}
```

$time\ complexity : O(nlog(n))$