

Data Structures and Algorithms Lab

02. Stacks and Queues. Lists.

Lab Code: 19ECSP201

Lab No: 02

Semester: III

Date: 23 Aug, 2019

Batch: C2

Question: Application of Stacks and Queues. Lists.

Objective: Understanding the usage of stacks and queues in real-time scenarios, as well, lists.

Problem 01:

The Hubballi city is definitely on the right path while talking towards city development. To name a few are road construction, city maintenance or building a drainage system, etc. Works are being carried out everywhere. Apart from these, good or bad, the city is also slowly adapting to technology and rural culture. Examples are U-Mall, Urban OASIS Mall, Laxmi Pride Cinemas, KFC, Dominos, etc., etc.

Urban Oasis Mall, is preparing a master plan on how to attract more customers and kind of shops that needs to be started in the remaining spaces of the mall. Also, it wants the process to get automated.



Fig: Urban OASIS Mall, Hubli City

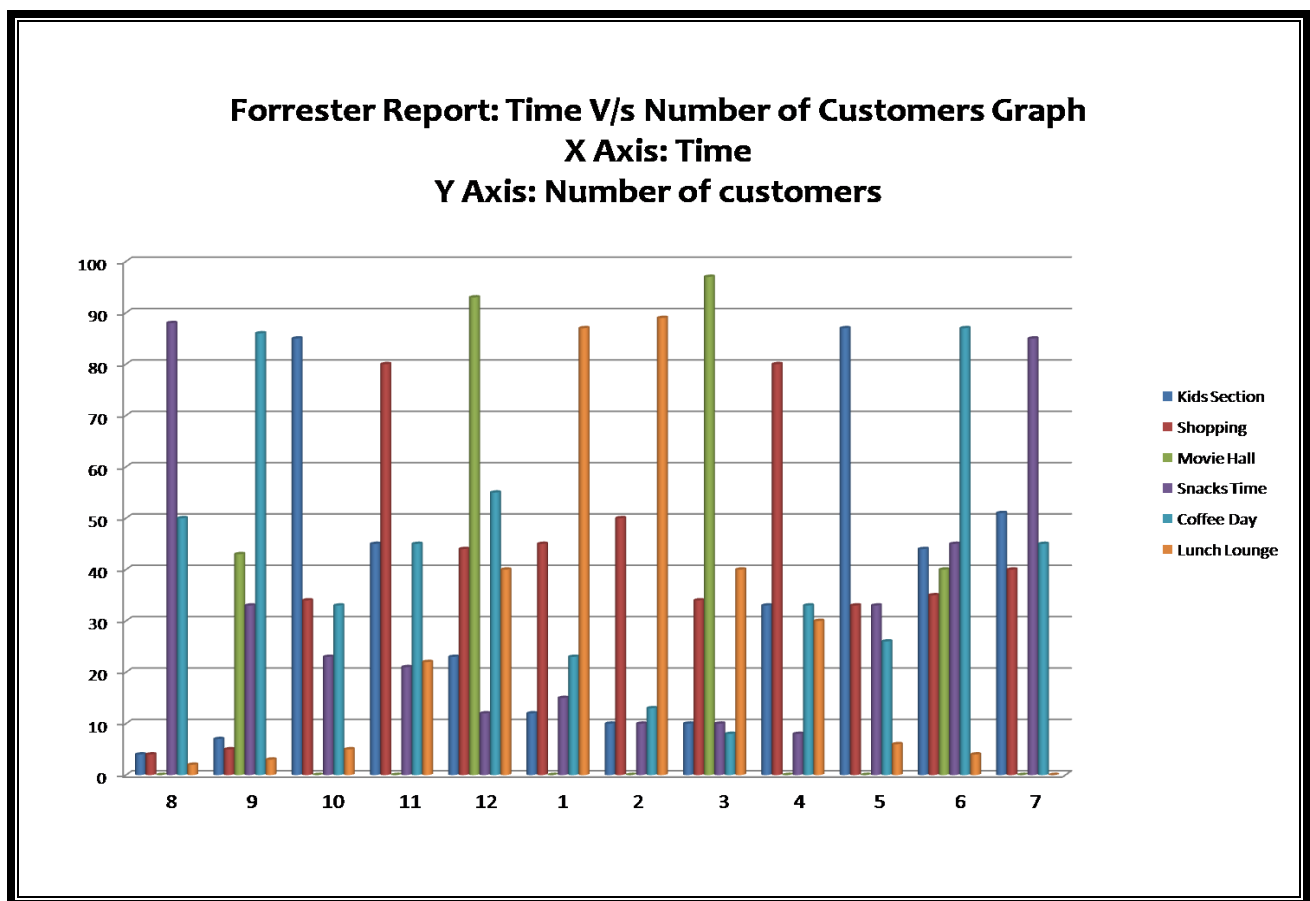
[Image Reference: ebharat.com]

DSA Lab 02: Application of Stacks and Queues. Lists.

As the first initiative, the mall wants to attract more customers. And the technique adapted is to send the messages every hour to all the mobiles detected in the nearby locality. (Spanning 1km radius)

It was necessary to know the user habits at other malls to decide what kind of message would be appropriate to be sent at every hour. The authorities had approached the **Forrester Report** to get the survey done!

Forrester Reports collected the stats for the number of customers present at each hour in 20 different malls of India. And after a study for 1 year, here is the **Forrester Report** summary:



According to the survey analysis, at each shop there are more customers at the specified timings:

Kids Section: 10.00 and 5.00

Shopping: 11.00 and 4.00

Movie Hall: 12.00 and 3.00

Snacks Time: 08.00 and 7.00

Coffee Day: 9.00 and 6.00

Lunch Lounge: 1.00 and 2.00

Here is the peak flow of customers at every hour:

8.00 - Snacks Time
9.00 – Coffee Day
10.00 – Kids Section
11.00 – Shopping
12.00 – Movie Hall
1.00 – Lunch Lounge
2.00 – Lunch Lounge
3.00 – Movie Hall
4. 00 – Shopping
5.00 – Kids Section
6.00 – Coffee Day
7.00 – Snacks Time

Do you observe any pattern above??

At Professor's Desk

After looking at the message pattern, **STACK** looks like the best suitable data structure for the application. But we will have to maintain two stacks. At every hour a pop from the stack will be a push to another stack. Can you justify why the stack is the best one? A little more help to you.

Use the below structures:

```
struct advertisement
{
    char message_sent[30];    // Mobile advertisement to be made
    int time[2];              // Time of broadcast
};

struct stack
{
    int top;                  // The stack top
    int active;               // Status to indicate if the stack is active
    struct advertisement items[STACKSIZE]; // Items tracking advertisements
};
```

typedef struct stack STACK;

Provide the functionalities in main for:

1. Display the message being broadcasted
2. Update Time
3. Print All Messages
4. Exit

Task Description:

1. **Display the message being broadcasted** – Peek into the active stack
2. **Update Time** - Pop the value from one stack and push it to another stack. When active stack becomes empty, make it inactive and activate another stack.
3. **Print All Messages** – Print both the stack contents clearly stating the type of stack
4. **Exit** – Exit from the menu

Initialize the data in stack statically before providing the above options to the user. Before the above menu is being printed to the user, the initialization function needs to be called, which will initialize one stack with respected content and keep the other empty.

Split the code into 3 different files. Test for all pre and postconditions. Some part of the code is already done for you. You need to do the remaining.

Wait!

That is not all of it.

There is another problem that needs to be solved.

Problem 02:

Gartner Report for this year has made an interesting prediction about the producer-consumer problem. Looking at the different advancements and researches and mainly motivated from nature, the report says:

“Producer is no more only a producer and so about the consumer”

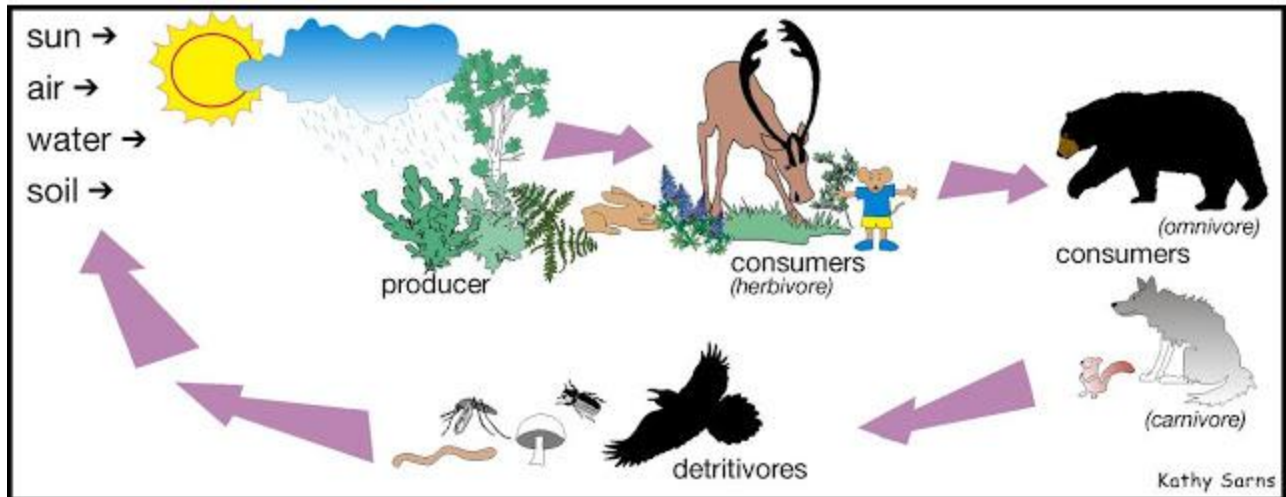


Fig: The Producer-Consumer Food chain

[Image Reference: chapeijin.blogspot.com]

Well,

For an engineer that is too much information. You can start coding!

At Professor's Desk

The problem has left behind many open things. Let us make all valid assumptions and move ahead with coding.

Our task is to simulate a producer which will produce as well as consume and so about the consumer.

Let us visualize and implement the new prediction.

Any doubts in why we are selecting a queue to implement it?

Use the below structures:

```
struct indices
{
    int front1;
    int rear1;
    int front2;
    int rear2;
    int produce;
    int consume;
};

struct prod_cons
{
    int data[MAXQUEUE];
    struct indices index;
};
```

typedef struct prod_cons PC;

Provide the functionalities in main for:

1. Produce in Producer
2. Consume from Consumer
3. Produce to Consumer using Producer
4. Consume from Producer
5. Print Producer-Consumer
6. Get the Active Producer and Consumer
7. Exit

Task Description:

1. **Produce In Producer** – Enqueue a random data into the producer queue
2. **Consume from Consumer** – Dequeue the data from consumer queue
3. **Produce in Consumer** – Enqueue the data into consumer from producer
4. **Consume from Producer** – Dequeue the data from producer queue and enqueue into consumer
5. **Print Producer Consumer** – Print the details present in both the queue with appropriate messages
6. **Get the Active Producer and Consumer** – Print in which queue the last enqueue and dequeue has happened
7. **Exit** – Exit from Menu

Points of Discussion:

- How are you going to visualize producer and consumer in integer array 'data'?
- What will be the empty condition?
- What will be the full condition?
- How will you generate a random integer data? Of what range?

Oh, snap! Wait!

There is another problem.

Problem 03:



Charlie, the new owner of the chocolate factory after returning from 'Oompa-Loompa' land has got a new recipe for making chocolates. Well, the old recipe was robbed out by a few workers, and the chocolate factory had to shut down as every cat and dog in city started making the same kind of chocolates. Charlie had to shut his factory and make an adventurous expedition in search of new recipe. That's when he ended up in Oompa-Loompa land. The Oompa-Loompas offered him with 2 new varieties in return of their favorite chow, beans. Now, Sssssssshhhh!!! One new secret recipe is shared only with you! Here is the procedure:

First, you need to buy the following:

200 KitKat

10 Silk

30 Bourneville

200 Wafers

30 Lint

Now you need to melt all in the same above order, one by one, whip it, grind it, bake it, freeze it and finally mold it to a dream fantasy chocolate.

Your task is to do it all. Go to market buy all the chocolates required, melt them in the same order and prepare the new chocolate.

Uno, Dos, Tres...

Let the choco-licious preparation begin!

At Professors Desk

Did you see how an interesting topic can be made exciting by asking someone to code? (Seriously??)

Your task is:

1. First, buy all the required variety of chocolates. We may not get all in the same shop. If the desired quantity is not available then we need to buy from next shop.

Maintain a linked list structure:

```
struct choc_fantasy
{
    char choc_name[20];
    int quantity;
    struct choc_fantasy * prev;
    struct choc_fantasy * next;
};
```

You will keep on calling the function “insert_at_end” until you buy all the required chocolates in specified quantity. The required chocolates can be purchased in any order.

2. Melt and prepare fantasy chocolate

Now melting process has to happen exactly in the same way as said. You will search for KitKat in linked list and delete the nodes (it could be one node having 200 quantities or split up in any number of nodes). Print appropriate message after adding each ingredient.

3. Then print the message “Secret Recipe Chocolate is now Ready” and exit the program.

**** Happy Coding ****