Stack and Queue

**Stack DataStructure:**

Stack is a type of datastructre where the element last entered in an array is going to first out. We can say it as LIFO (Last In First Out) or FILO(First In Last Out) . এর মানে হচ্ছে এই ডেটাস্ট্রাকচার এ একটি array তে যে element আগে প্রবেশ করবে সে element টা সবার শেষ এ বের হবে এবং যে element সবার পরে প্রবেশ করবে সে element সবার আগে প্রবেশ করবে ।

এই ডেটাস্ট্রাকচার এ কয়েক টি বিষয় আছে ,

* push() – for adding element to the last
* pop() – for removing element from the last
* top()- the current top element in the arrey
* empty() – is the array is empty or not



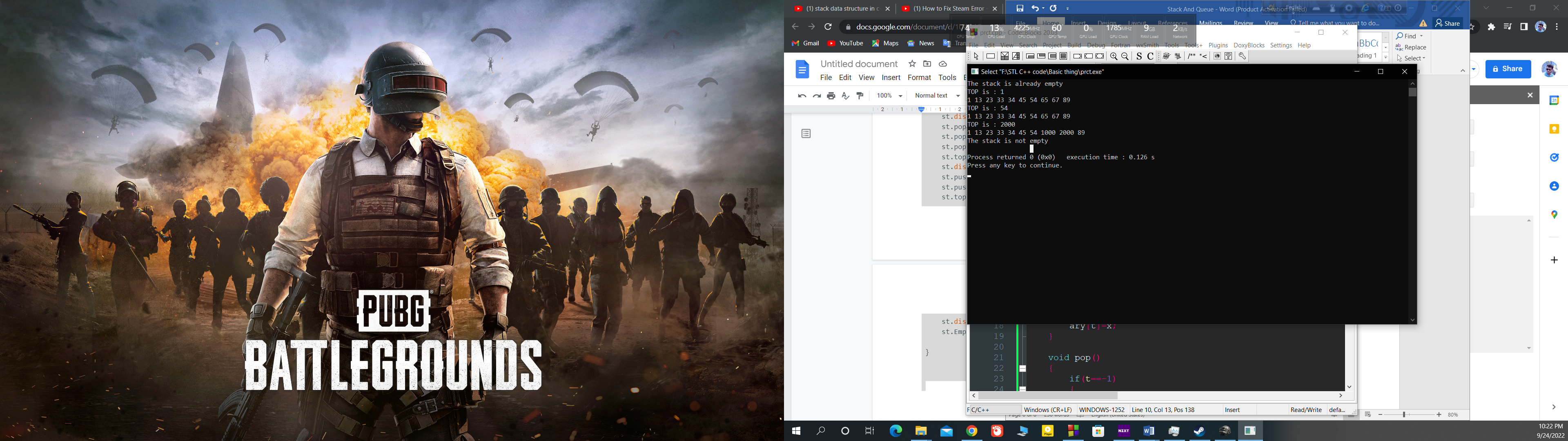
How stack works :



Code:

|  |  |
| --- | --- |
| |  | | --- | | #include<bits/stdc++.h> using namespace std; #define size 10 class Stack {     int t=-1;     int ary[size];  public:     void push( int x)     {         if(t>=size)         {             cout<<"The stack is overflow "<<endl;             return;         }         t++;         ary[t]=x;     }      void pop()     {         if(t==-1)         {             cout<<"The stack is already empty"<<endl;             return;         }         t--;     }     void top()     {         if(t==-1)         {             cout<<"The stack is already empty"<<endl;             return;         }         cout<<"TOP is : "<<ary[t]<<endl;     }      void Empty()     {          if(t==-1)         {             cout<<"The stack is already empty"<<endl;             return;         }         else         {             cout<<"The stack is not empty "<<endl;         }     }      void display()     {          for(auto u:ary)         {             cout<<u<<" ";         }         cout<<endl;     }    };  int main() {     Stack st;     st.Empty();     st.push(1);     st.top();     st.push(13);     st.push(23);     st.push(33);     st.push(34);     st.push(45);     st.push(54);     st.push(65);     st.push(67);     st.push(89);     st.display();     st.pop();     st.pop();     st.pop();     st.top();     st.display();     st.push(1000);     st.push(2000);     st.top();     st.display();     st.Empty();  } | |

OUTPUT:



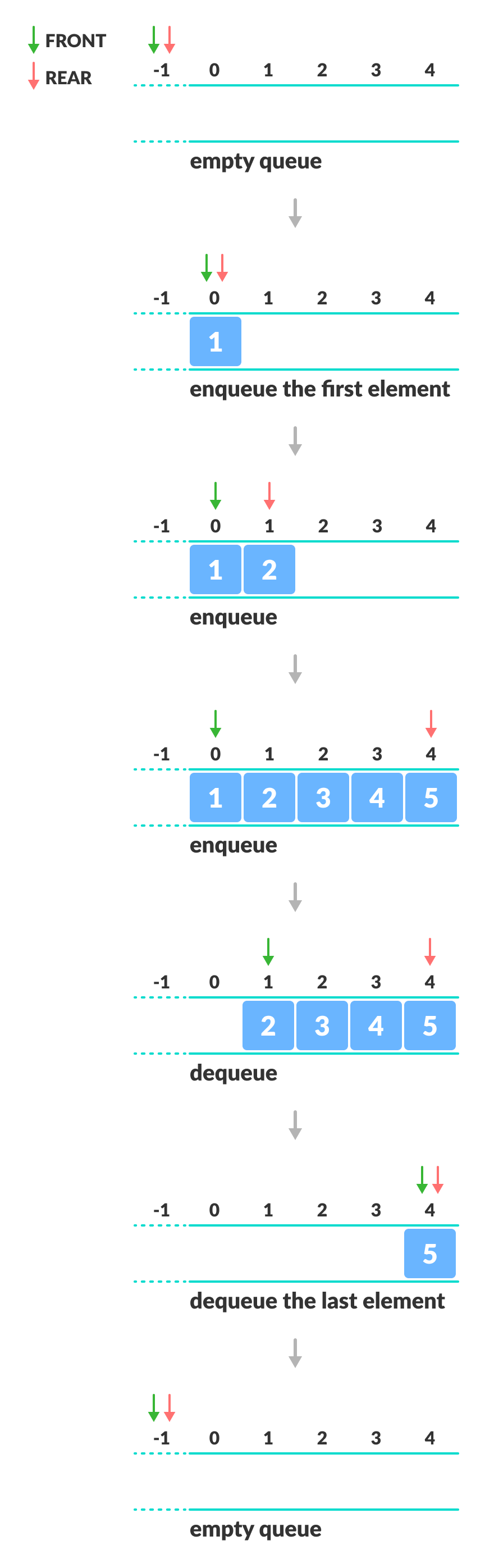
**Queue Datastructure :**

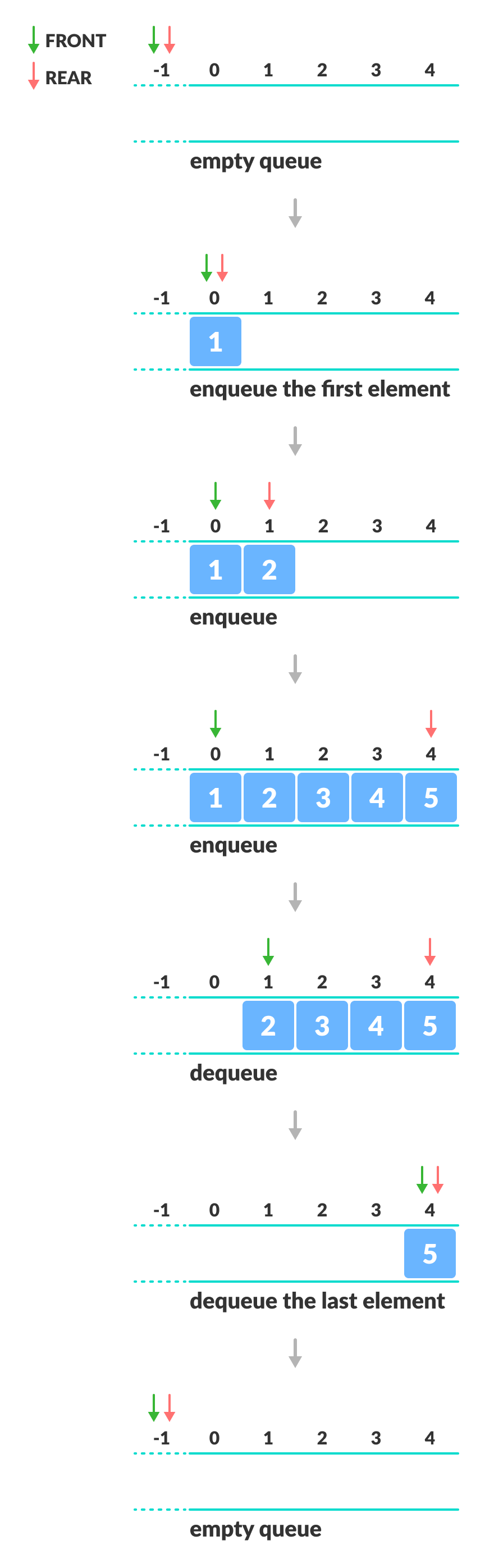
Queue is a kind of datastructure where we arrange data by following FIFO(First In First Out). We can imagine it as a ticket counter where who goes first get the ticket first and out of the line first .

There are some operations same as stack:

* Enqueue : same as push. Adding some element in an array
* Dequeue: same as pop. Deleting element from the right side of the arrey
* isEmpty: is the array is empty or not?
* isFull: check if the array is full or not ?
* peek : get the value in the front without removing it

Operations of queue like this:





Code:

|  |
| --- |
| #include<bits/stdc++.h> using namespace std; #define SIZE 5 #define end "\n" class Queue { private:     int f,b,ary[SIZE]; public:     Queue() //making a constructor     {         f=-1;         b=-1;     }     bool isFull()     {         if(f==0   &&  b==(SIZE-1))         {             return true;         }         return false;     }      bool isEmpty()     {         if(f == -1)         {             return true;         }         return false;     }      void push( int x)     {         if(isFull())         {             cout<<"Queue is overFlow "<<endl;          }         else         {             if(f==-1)             {                 f=0;             }             b++;             ary[b]=x;              cout<<"element inserted successfully"<<endl;          }      }      void pop()     {         if(isEmpty())         {             cout<<"Queue is already empty"<<endl;         }          else         {             if(f>=b)             {                 f=-1;                 b=-1;                 return;             }             else             {                 f++;                 cout<<"Deleted data successfully"<<endl;                 return;             }          }     }      void top()     {         if(isEmpty())         {             cout<<"Queue is already empty"<<endl;             return;         }         else         {             cout<<"TOP : "<<ary[f]<<endl;             return;         }     }      void Empty()     {         if(isEmpty())         {             cout<<"Queue is empty  "<<endl;         }         else         {             cout<<"Queue is not empty  "<<endl;         }     }      void display()     {         if(isEmpty())         {             cout<<"Queue is empty"<<endl;         }         else         {             cout<<endl<<endl;             cout<<"Showing output : "<<endl;              for(int i=f; i<=b; i++)             {                 cout<<ary[i]<<" ";              }             cout<<endl<<endl;          }      }  };  int main() {     Queue q;     q.pop();     q.push(1);     q.push(2);     q.push(3);     q.push(4);     q.push(5);  //showing "Queue is overflow "     q.push(6);  //showing display     q.display();      q.pop();     q.display();  //pop five times     q.pop();      q.pop();      q.pop();      q.pop();      q.pop();      q.display(); //again push 5 element     q.push(10);     q.push(20);     q.push(30);     q.push(40);     q.push(50);     q.display();  //delete one element     q.pop();  //showing output     q.display(); } |

Output: copy the code and run in the codeblocks