1.INSERT ELEMENTS IN A QUEUE

```
#include <iostream>
#include <queue>

using namespace std;

int main() {

  // create a queue of string
  queue<string> animals;

  // push elements into the queue
  animals.push("Cat");
  animals.push("Dog");

  cout << "Queue: ";

  // print elements of queue
  // loop until queue is empty
  while(!animals.empty()) {
    // print the element
    cout << animals.front() << ", ";

    // pop element from the queue
    animals.pop();
  }

  cout << endl;

  return 0;
}
```

2.Remove elements from queue

```
#include <iostream>
#include <queue>
using namespace std;

// function prototype for display_queue
```

```
utility
void display_queue(queue<string> q);

int main() {

  // create a queue of string
  queue<string> animals;

  // push element into the queue
  animals.push("Cat");
  animals.push("Dog");
  animals.push("Fox");

  cout << "Initial Queue: ";
  display_queue(animals);

  // remove element from queue
  animals.pop();

  cout << "Final Queue: ";
  display_queue(animals);
  return 0;
}

// utility function to display queue
void display_queue(queue<string> q) {
  while(!q.empty()) {
    cout << q.front() << ", ";
    q.pop();
  }

  cout << endl;
}
```

3.Access Element from the Queue

We can access the elements of
a queue using the following methods:

- front() - returns the element from the front of the queue

- back() - returns the element from the back of the queue

Example:

```
#include <iostream>
#include <queue>
using namespace std;

int main() {

  // create a queue of int
  queue<int> nums;

  // push element into the queue
  nums.push(1);
  nums.push(2);
  nums.push(3);
  // get the element at the front
  int front = nums.front();
  cout << "First element: " << front << endl;

  // get the element at the back
  int back = nums.back();
  cout << "Last element: " << back << endl;

  return 0;
}
```

Get the size of the Queue

We use the size() method to get the number of elements in the queue. For example,

```
#include <iostream>
#include <queue>
using namespace std;

int main() {

  // create a queue of string
  queue<string> languages;

  // push element into the queue
  languages.push("Python");
  languages.push("C++");
  languages.push("Java");
  // get the size of the queue
  int size = languages.size();
  cout << "Size of the queue: " <<
size;

  return 0;
}
```

Check if the Queue is Empty

We use the empty() method to check if the queue is empty. This method returns:

- **1 (true)** - if the queue is empty

- **0 (false)** - if the queue is not empty

For example,

```
#include <iostream>
#include <queue>
using namespace std;

int main() {

  // create a queue of string
  queue<string> languages;

  cout << "Is the queue empty? ";
  // check if the queue is empty
  if (languages.empty()) {
    cout << "Yes" << endl;
  }
  else {
    cout << "No" << endl;
  }

  cout << "Pushing elements..." <<
endl;

  // push element into the queue
  languages.push("Python");
  languages.push("C++");

  cout << "Is the queue empty? ";
  // check if the queue is empty
  if (languages.empty()) {
    cout << "Yes";
  }
  else {
    cout << "No";
  }

  return 0;
}
```