Q1. Print all elements of a queue in C++ STL

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

int main() {
    queue<int> s;

    s.push(1);
    s.push(2);
    s.push(3);
    s.push(4);
    s.push(5);
    s.push(6);
    s.push(7);

    while (!s.empty()) {
        cout << s.front() << " ";
        s.pop();
    }

    return 0;
}
```

Q2. Create a railways ticket counter and allot ticket to first come first take using c++ STL

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

int main()
{
   queue <int> q;

   int ch = 0, i = 1;

   while(ch != 3)
   {
      cout<<"1. Insert cumstomer"<<endl;
      cout<<"2. Give ticket"<<endl;
      cout<<"3. Exit"<<endl<<endl;

      cout<<"choice = ";
      cin>>ch;

      switch(ch)
      {
         case 1:
            {
               q.push(i);
               i = i + 1;

               break;
            }
         case 2:
            {
               if(!q.empty())
               {
                  cout<<"tickte gave to customer = "<<q.front()<<endl;
                  q.pop();
               }
               else
                  cout<<"queue is empty"<<endl;

               break;
            }
         case 3:
            {
               exit(0);
               break;
            }
      }
   }
}
```

Q3. Check if a queue is empty or not using queue::size() function

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

using namespace std;

int main()
{
   queue <int> q;

   if(q.size())
   {
      cout<<"queue is not empty";
   }
   else
   {
      cout<<"queue is empty";
   }

}
```

Q4. Manage a queue for multiple input and store in ascending order of his first character

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

int main()
{
   queue <string> q;
   vector <string> v;

   int n = 0, prv = 0, nxt = 0;
   string s, s1, s2;

   cout<<"How many strings are you input = ";
   cin>>n;

   for(int i = 0; i < n; i++)
   {
      cout<<i<<") ";
      cin>>s;

      v.push_back(s);
      cin.ignore();
   }

   for(int i = 0; i < v.size(); i++)
   {
      for(int j = i + 1; j < v.size(); j++)
      {
         if(v.at(i) <= v.at(j))
         {
            continue;
         }
         else
         {
            string tmp = v.at(i);
            v.at(i) = v.at(j);
            v.at(j) = tmp;
         }
      }
   }

   for(int i = 0; i < v.size(); i++)
   {
      q.push(v.at(i));
   }

   while(!q.empty())
   {
      cout<<q.front()<<endl;
      q.pop();
   }
}
```

Q5. Schedule a interview by using applicant's reaching time using c++ STL

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

int main()
{
   queue <float> q;
   vector <float> v = {10.30, 7.00, 8.45, 12.00, 9.15};

   for(int i = 0; i < v.size(); i++)
   {
      for(int j = i + 1; j < v.size(); j++)
      {
         if(v.at(i) < v.at(j))
         {
            continue;
         }
         else
         {
            float tmp = v.at(i);
            v.at(i) = v.at(j);
            v.at(j) = tmp;
         }
      }
   }

   while(!v.empty())
   {
      q.push(v.front());
      v.erase(v.begin());
   }

   while(!q.empty())
   {
      cout<<q.front()<<endl;
      q.pop();
   }
}
```

Q6. In C++ STL, Queue is a type of container that follows FIFO (First-in-First-Out) elements arrangement i.e. the elements which insert first will be removed first. In the queue, elements are inserted at one end known as "back" and are deleted from another end known as "front".empty() and size() function of the queue with the Example.

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

int main() {
    queue<int> q;

    // Initially queue is empty
    if (q.empty()) {
        cout << "Queue is empty\n";
    }

    // Inserting elements
    q.push(10);
    q.push(20);
    q.push(30);

    // Now queue is not empty
    if (!q.empty()) {
        cout << "Queue is not empty\n";
    }

    // Size of queue
    cout << "Size of queue: " << q.size() << endl;

    // Let's remove one element
    q.pop();

    // New size
    cout << "Size after one pop: " << q.size() << endl;

    return 0;
}
```

Q7. Exchange the contents of two queues but the queues must be of the same data type, although sizes may differ.

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

int main()
{
   queue <int> q1, q2;

   for(int i = 1; i <= 5; i++)
   {
      q1.push(i*100);
   }

   for(int i = 1; i <= 5; i++)
   {
      q2.push(i*1000);
   }

   swap(q1, q2);

   cout<<"q1 = ";
   while(!q1.empty())
   {
      cout<<q1.front()<<" ";
      q1.pop();
   }

   cout<<endl<<"q2 = ";
   while(!q2.empty())
   {
      cout<<q2.front()<<" ";
      q2.pop();
   }

}
```

Q8. Insert a new element into the queue container, the new element is added to the end of the queue

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

int main()
{
   queue <int> v;

   v.push(10);
   v.push(999);
}
```

Q9. Adds the element ‘g’ at the end of the queue.

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

int main() {
    queue<char> q;


    q.push('a');
    q.push('b');
    q.push('c');


    q.push('g');

    while (!q.empty()) {
        cout << q.front() << " ";
        q.pop();
    }

    return 0;
}
```

Q10. Deletes the first element of the queue.

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

int main() {
    queue<int> q;

    q.push(10);
    q.push(20);
    q.push(30);

    cout << "Before pop, front = " << q.front() << endl;

    // Remove the first element
    q.pop();

    cout << "After pop, front = " << q.front() << endl;

    return 0;
}
```