ASSIGNMENT 01

Q1). A) Use the stack abstract data type to solve the Parenthesis Matching Problem, where the input is the string containing parentheses ((,), {, }, [,]), and the goal is to determine if the parentheses are balanced.

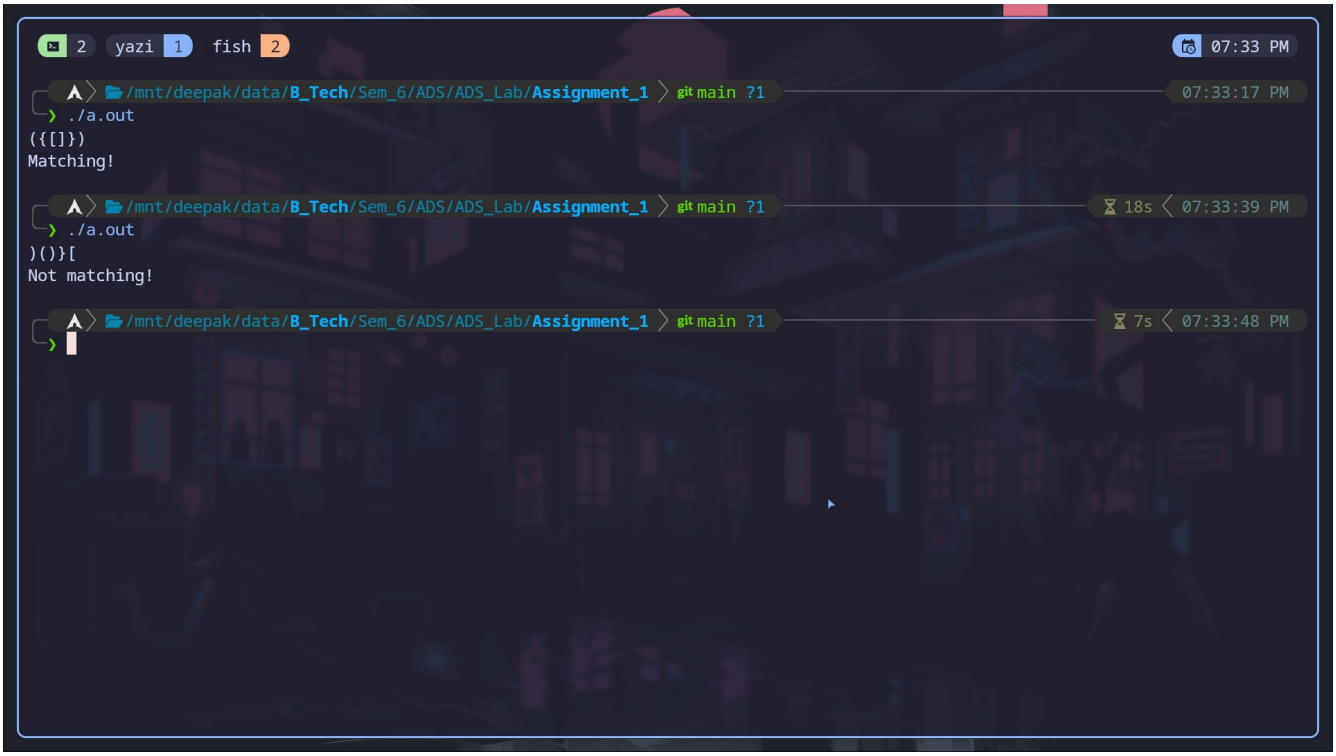
CODE:

```
#include <iostream>

using namespace std;

class Stack {
private:
  int top;
  char arr[100];

public:
  Stack() { top = -1; }

  void push(char ch) {
    if (top >= 99) {
      cout << "Stack Overflow" << endl;
      return;
    }
    arr[++top] = ch;
    return;
  }

  void pop() {
    if (top < 0) {
      cout << "No Elements to pop" << endl;
      return;
```

```
    }
    top--;
    return;
  }

  char topelem() { return arr[top]; }
  bool empty() {
    if (top < 0) {
      return true;
    }
    return false;
  }
};

int main() {
  string ins = "";
  cin >> ins;
  Stack s;

  for (int i = 0; i < ins.length(); i++) {
    if (ins[i] == '(' || ins[i] == '{' || ins[i] == '[') {
      s.push(ins[i]);
    } else if (ins[i] == ')' && !s.empty() && s.topelem() == '(') {
      s.pop();
    } else if (ins[i] == '}' && !s.empty() && s.topelem() == '{') {
      s.pop();
    } else if (ins[i] == ']' && !s.empty() && s.topelem() == '[') {
      s.pop();
    } else {
      cout << "Not matching!" << endl;
      return 0;
    }
  }

  if (s.empty()) {
    cout << "Matching!" << endl;
```

```
  } else {
    cout << "Not matching!" << endl;
  }

  return 0;
}
```

OUTPUT:



B) Name the data structure used to solve the Parenthesis Matching Problem.
Ans→ Stack Data Structure.

Q2). A) Design a ticket booking system where customers arrive to book tickets and join a queue. The system should process customers in a first-come, first-served (FIFO) manner. The queue must provide the following operations (options) to the user:

1. Add Customer -> Add a new customer to the booking queue.

2. Process Booking -> Process the ticket booking for the current customer in line and remove them from the queue after processing.

3. View Queue -> Display the current list of customers waiting in the queue.

The system should simulate real-life ticket booking, where customers are served in the order of arrival.

CODE:

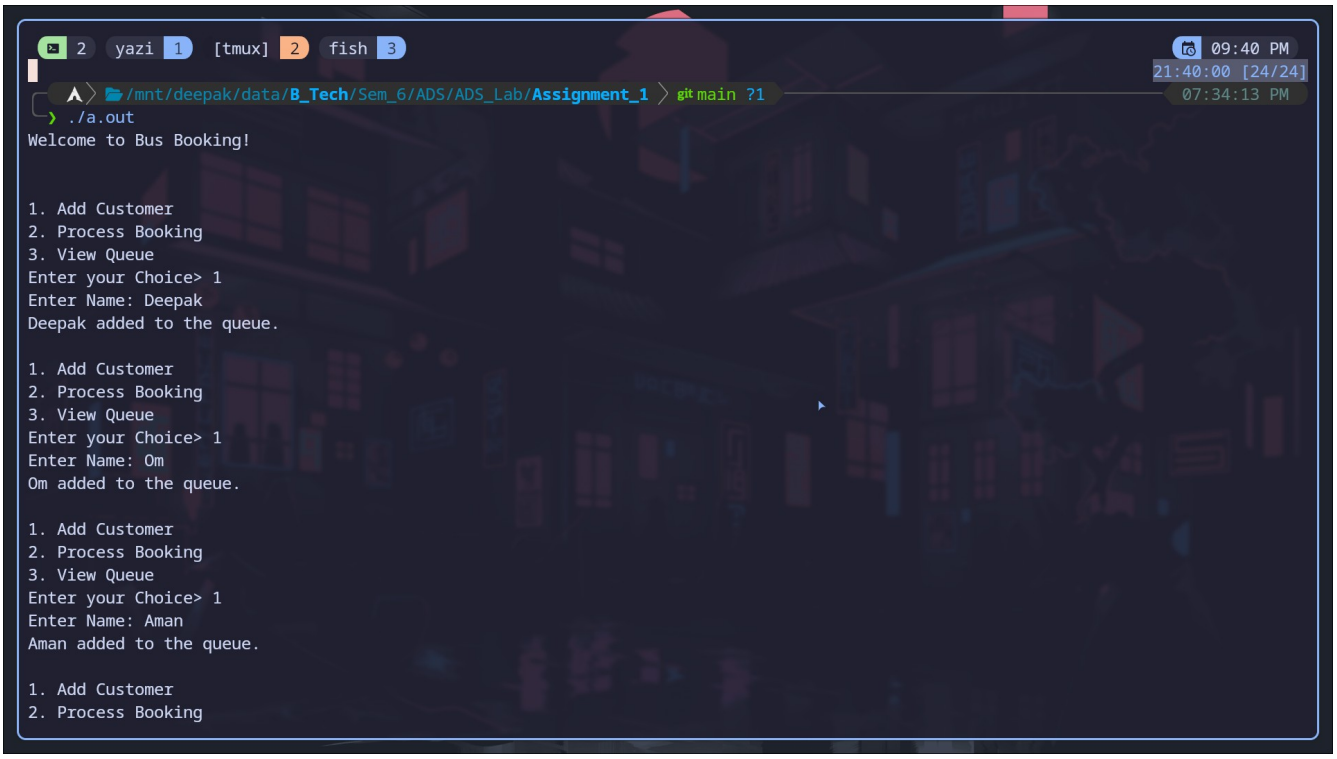
```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Queue {
private:
  int front, rear;
  vector<string> namearr;

public:
  Queue() : front(-1), rear(-1) {}

  void push(const string &name) {
    if (front == -1) {
      front = 0;
    }
```

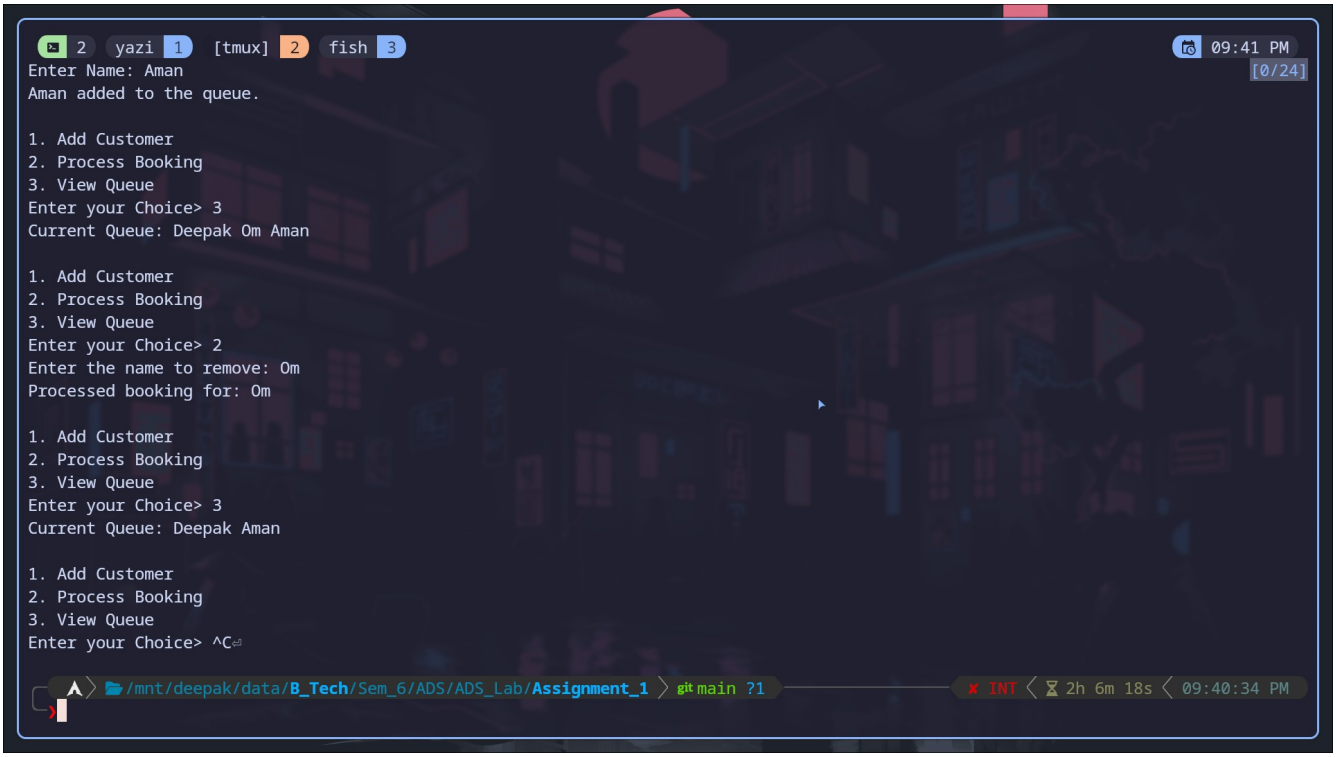
```
  rear++;
  if (rear < namearr.size()) {
    namearr[rear] = name;
  } else {
    namearr.push_back(name);
  }
  cout << name << " added to the queue.\n";
}

void pop() {
  if (empty()) {
    cout << "Queue Underflow\n";
    return;
  }
  front++;
  if (front > rear) {
    front = rear = -1;
  }
}

string top() const {
  if (empty()) {
    return "Queue is empty!";
  }
  return namearr[front];
}

bool empty() const { return front == -1 || front > rear; }

void remove(const string &name) {
  bool found = false;
  vector<string> newQueue;
  for (int i = front; i <= rear; ++i) {
    if (namearr[i] == name && !found) {
      found = true;
    } else {
```

```
        newQueue.push_back(namearr[i]);
      }
    }
    if (found) {
      cout << "Processed booking for: " << name << "\n";
    } else {
      cout << "Booking for " << name << " not found.\n";
    }

    namearr = newQueue;
    front = newQueue.empty() ? -1 : 0;
    rear = newQueue.size() - 1;
  }

  void display() const {
    if (empty()) {
      cout << "Queue is empty!\n";
      return;
    }
    cout << "Current Queue: ";
    for (int i = front; i <= rear; ++i) {
      cout << namearr[i] << " ";
    }
    cout << "\n";
  }
};

int main() {
  Queue q;
  int choice;

  cout << "Welcome to Bus Booking!\n\n";

  while (true) {
    cout << "\n1. Add Customer\n2. Process Booking\n3. View Queue\nEnter
your "
```

```
            "Choice> ";
    cin >> choice;

    if (choice == 1) {
      cout << "Enter Name: ";
      string name;
      cin >> name;
      q.push(name);
    } else if (choice == 2) {
      cout << "Enter the name to remove: ";
      string name;
      cin >> name;
      q.remove(name);
    } else if (choice == 3) {
      q.display();
    } else {
      cout << "Invalid choice! Try again.\n";
    }
  }

  return 0;
}
```

OUTPUT:


2 yazi 1 [tmux] 2 fish 3
09:40 PM
21:40:00 [24/24]
/mnt/deepak/data/B_Tech/Sem_6/ADS/ADS_Lab/Assignment_1 git main ?1
07:34:13 PM
./a.out
Welcome to Bus Booking!
1. Add Customer
2. Process Booking
3. View Queue
Enter your Choice> 1
Enter Name: Deepak
Deepak added to the queue.
1. Add Customer
2. Process Booking
3. View Queue
Enter your Choice> 1
Enter Name: Om
Om added to the queue.
1. Add Customer
2. Process Booking
3. View Queue
Enter your Choice> 1
Enter Name: Aman
Aman added to the queue.
1. Add Customer
2. Process Booking


2 yazi 1 [tmux] 2 fish 3
09:41 PM
[0/24]
Enter Name: Aman
Aman added to the queue.
1. Add Customer
2. Process Booking
3. View Queue
Enter your Choice> 3
Current Queue: Deepak Om Aman
1. Add Customer
2. Process Booking
3. View Queue
Enter your Choice> 2
Enter the name to remove: Om
Processed booking for: Om
1. Add Customer
2. Process Booking
3. View Queue
Enter your Choice> 3
Current Queue: Deepak Aman
1. Add Customer
2. Process Booking
3. View Queue
Enter your Choice> ^C
/mnt/deepak/data/B_Tech/Sem_6/ADS/ADS_Lab/Assignment_1 git main ?1
INT 2h 6m 18s 09:40:34 PM

B) Name the data structure used for the ticket booking system.
Ans → Queue Data Structure.