



# **Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Summer,Year:2024),B.Sc.in CSE (Day)**

**Project**

**Course Title: Data Structures**

**Course Code: CSE105**

**Section:231-D1**

**Project Title: Airport Luggage Handling System**

## **Student Details**

<b>Name</b>		<b>ID</b>
<b>1.</b>	<b>MD.SHAJALAL</b>	<b>223002088</b>

**Project Date : 21-05-2024**

**Submission Date : 09-06-2024**

**Course Teacher's Name : Prof. Dr. Md. Saiful Azad**

## **Project Status**

**Marks: .....**

**Comments:.....**

**Signature:.....**

**Date:.....**

# Airport Luggage Handling System

## Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Luggage {
    char size[100];
    float weight;
    struct Luggage* next;
} Luggage;

typedef struct Passenger {
    char name[50];
    char flightDetails[50];
    int numOfBags;
    Luggage* luggageList;
    struct Passenger* next;
} Passenger;

typedef struct {
    Passenger* front;
    Passenger* rear;
} PassengerQueue;

typedef struct {
    Luggage* top;
    int maxCapacity;
    int currentSize;
} LuggageStack;
```

```

void enqueuePassenger(PassengerQueue* queue, char* name, char* flightDetails,
int numOfBags) {
    Passenger* newPassenger = (Passenger*)malloc(sizeof(Passenger));
    strcpy(newPassenger->name, name);
    strcpy(newPassenger->flightDetails, flightDetails);
    newPassenger->numOfBags = numOfBags;
    newPassenger->luggageList = NULL;
    newPassenger->next = NULL;

    if (queue->rear == NULL) {
        queue->front = queue->rear = newPassenger;
        return;
    }

    queue->rear->next = newPassenger;
    queue->rear = newPassenger;
}

Passenger* dequeuePassenger(PassengerQueue* queue) {
    if (queue->front == NULL) {
        return NULL;
    }

    Passenger* temp = queue->front;
    queue->front = queue->front->next;

    if (queue->front == NULL) {
        queue->rear = NULL;
    }

    return temp;
}

```

```

void addLuggage(Passenger* passenger, char* size, float weight) {
    Luggage* newLuggage = (Luggage*)malloc(sizeof(Luggage));
    strcpy(newLuggage->size, size);
    newLuggage->weight = weight;
    newLuggage->next = NULL;

    if (passenger->luggageList == NULL) {
        passenger->luggageList = newLuggage;
    } else {
        Luggage* temp = passenger->luggageList;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newLuggage;
    }
}

void pushLuggage(LuggageStack* stack, Luggage* luggage) {
    if (stack->currentSize >= stack->maxCapacity) {
        printf("Cart is full. Cannot add more luggage.\n");
        return;
    }

    luggage->next = stack->top;
    stack->top = luggage;
    stack->currentSize++;
}

Luggage* popLuggage(LuggageStack* stack) {
    if (stack->top == NULL) {
        return NULL;
    }
    Luggage* temp = stack->top;
    stack->top = stack->top->next;
    stack->currentSize--;
    return temp;
}

```

```

void initializeQueue(PassengerQueue* queue) {
    queue->front = queue->rear = NULL;
}

void initializeStack(LuggageStack* stack, int maxCapacity) {
    stack->top = NULL;
    stack->maxCapacity = maxCapacity;
    stack->currentSize = 0;
}

void displayQueue(PassengerQueue* queue) {
    Passenger* temp = queue->front;
    while (temp != NULL) {
        printf("Passenger: %s, Flight: %s, Number of Bags: %d\n", temp->name,
temp->flightDetails, temp->numOfBags);
        temp = temp->next;
    }
}

void cleanupLuggageList(Luggage* luggageList) {
    while (luggageList != NULL) {
        Luggage* temp = luggageList;
        luggageList = luggageList->next;
        free(temp);
    }
}

void cleanupQueue(PassengerQueue* queue) {
    while (queue->front != NULL) {
        Passenger* temp = dequeuePassenger(queue);
        cleanupLuggageList(temp->luggageList);
        free(temp);
    }
}

```

```
int main() {
    PassengerQueue queue;
    initializeQueue(&queue);

    LuggageStack cart;
    initializeStack(&cart, 10); // Assuming each cart can hold 10 items

    int choice;
    char name[50], flightDetails[50], size[100];
    int numOfBags;
    float weight;

    while (1) {
        printf("\nAirport Luggage Handling System\n");
        printf("1. Add Passenger to Check-in Queue\n");
        printf("2. Process Passenger Check-in\n");
        printf("3. Load Luggage onto Baggage Cart\n");
        printf("4. Unload Luggage onto Airplane\n");
        printf("5. Display Passenger Queue\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter Passenger Name: ");
                scanf("%s", name);
                printf("Enter Flight Details: ");
                scanf("%s", flightDetails);
                printf("Enter Number of Bags: ");
                scanf("%d", &numOfBags);
                enqueuePassenger(&queue, name, flightDetails, numOfBags);
                break;
```

case 2:

```
{ Passenger* p = dequeuePassenger(&queue);
  if (p != NULL)
  {
    for (int i = 0; i < p->numOfBags; i++) {
      printf("Enter Luggage Size: ");
      scanf("%s", size);
      printf("Enter Luggage Weight: ");
      scanf("%f", &weight);
      addLuggage(p, size, weight);
    }
    Luggage* luggage = p->luggageList;
    while (luggage != NULL) {
      Luggage* nextLuggage = luggage->next;
      pushLuggage(&cart, luggage);
      luggage = nextLuggage;
    }
    free(p);
  } else {
    printf("No passengers in the queue.\n");
  }
}
break;
```

case 3:

```
{
  Passenger* p = dequeuePassenger(&queue);
  if (p != NULL) {
    Luggage* luggage = p->luggageList;
    while (luggage != NULL) {
      Luggage* nextLuggage = luggage->next;
      pushLuggage(&cart, luggage);
      luggage = nextLuggage;
    }
    free(p);
  } else {
```

```

        printf("No passengers in the queue.\n");
    }
}
break;
case 4:
    printf("\nLoading luggage onto the airplane:\n");
    Luggage* loadedLuggage;
    while ((loadedLuggage = popLuggage(&cart)) != NULL) {
        printf("Loaded luggage of size %s and weight %.2f onto the
airplane.\n", loadedLuggage->size, loadedLuggage->weight);
        free(loadedLuggage);
    }
    break;
case 5:
    printf("Passenger Queue:\n");
    displayQueue(&queue);
    break;
case 6:
    cleanupQueue(&queue);
    printf("Exiting system.\n");
    return 0;
default:
    printf("Invalid choice. Please try again.\n");
}
}

return 0;
}

```



## Output

```
F:\Programming\C_programr  X  +  v

Airport Luggage Handling System
1. Add Passenger to Check-in Queue
2. Process Passenger Check-in
3. Load Luggage onto Baggage Cart
4. Unload Luggage onto Airplane
5. Display Passenger Queue
6. Exit
Enter your choice:
```

```
F:\Programming\C_programr  X  +  v

Airport Luggage Handling System
1. Add Passenger to Check-in Queue
2. Process Passenger Check-in
3. Load Luggage onto Baggage Cart
4. Unload Luggage onto Airplane
5. Display Passenger Queue
6. Exit
Enter your choice: 1
Enter Passenger Name: Md.Shajalal
Enter Flight Details: CSE
Enter Number of Bags: 22
```

## Airport Luggage Handling System

1. Add Passenger to Check-in Queue
2. Process Passenger Check-in
3. Load Luggage onto Baggage Cart
4. Unload Luggage onto Airplane
5. Display Passenger Queue
6. Exit

Enter your choice: 1

Enter Passenger Name: Md.Shajalal

Enter Flight Details: CSE

Enter Number of Bags: 22

## Airport Luggage Handling System

1. Add Passenger to Check-in Queue
2. Process Passenger Check-in
3. Load Luggage onto Baggage Cart
4. Unload Luggage onto Airplane
5. Display Passenger Queue
6. Exit

Enter your choice: 2

Enter Luggage Size: 11

Enter Luggage Weight: 20