Summer 2025 Final Exam

Q1: Delete the Middle Element (Stack ADT) - 33.3 Points

Task

Using the provided **Stack ADT**, remove the **middle element** from a stack **without using any extra data structure.**

- Middle index (1-based from **bottom**): `floor((n + 1) / 2)`.
- You may use **recursion** (call stack is allowed).
- Output/printing shows the stack **from top to bottom**.

Write

```
void delete_middle(Stack *s);
```

Rules

- Use only the ADT operations (`stack_init`, `stack_push`, `stack_pop`, `stack_peek`, `stack_empty`).

Given (stack.h)

```
#ifndef STACK_H
#define STACK_H
typedef struct { int data[1000]; int top; } Stack;
void stack_init(Stack *s);
int  stack_push(Stack *s, int v);     // returns 1 on success, 0 on fail
int  stack_pop(Stack *s, int *out);   // returns 1 on success, 0 on empty
int  stack_peek(const Stack *s, int *out);
int  stack_empty(const Stack *s);     // 1 if empty, else 0
#endif
```

Examples (top → bottom)

- Input: `[50, 40, 30, 20, 10]` → remove `30` → Output: `[50, 40, 20, 10]`
- Input: `[40, 30, 20, 10]` → remove `20` → Output: `[40, 30, 10]`

Constraints & Hints

- Let `n` be the stack size. Aim for **O(n)** time.
- Recursion approach: pop until you reach the middle, skip it, then push back the rest.
- If `n ≤ 1`, the result is either empty or unchanged.

Q2: Flight Schedule Sorter - 33.33 Points

An airline company needs a program to help sort its daily flight schedule. Each flight record contains:

- Flight number (e.g., `"AA123"`)
- Destination city (no spaces)
- Departure time (in 24-hour `HH:MM` format)

Your task:

1. Read `N` flights from standard input.
2. Store them in a **dynamically allocated array** of a `Flight` **struct**.
3. **Bubble sort** the flights by departure time (earliest first). If times are the same, sort by flight number alphabetically.
4. Print the sorted schedule.

Function Signature:

```
Flight* readFlightsFromFile(char* filename, int* n);
void sortFlights(Flight* flights, int n);
void printFlightSchedule(Flight* flights, int n);
```

Flight struct:

```
typedef struct {
    char flightNumber[10];
    char destination[50];
    char departureTime[6]; // HH:MM format
} Flight;
```

input.txt

```
8
AA123 Dallas 14:30
BA200 London 09:45
DL450 Atlanta 14:30
UA999 Chicago 06:15
AF320 Paris 22:10
QF101 Sydney 05:50
JL720 Tokyo 09:45
LH400 Frankfurt 14:05
```

Q3: Bank Customer Service Simulation - 33.33 Points

A bank uses a **queue** to serve customers in the order they arrive. Each customer record contains:

- Ticket number (integer)

- Name (string without spaces)
- Service time in minutes (integer)

Your task:

1. Implement a **Queue ADT** using a fixed-size array.
2. Read `N` customers from input and **enqueue** them in the order given.
3. **Dequeue** customers one by one and print:
    - Ticket number
    - Name
    - Time they will be served (cumulative sum of service times so far)

Given (queue.h)

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE_SIZE 100

typedef struct {
    int ticket;
    char name[50];
    int serviceTime;
} Customer;

typedef struct {
    Customer data[MAX_QUEUE_SIZE];
    int front;
    int rear;
    int size;
} Queue;

void queue_init(Queue *q);
int queue_enqueue(Queue *q, Customer c);  // returns 1 on success, 0 on fail
int queue_dequeue(Queue *q, Customer *out); // returns 1 on success, 0 on empty
int queue_front(Queue *q, Customer *out);  // returns 1 on success, 0 on empty
int queue_empty(Queue *q);              // returns 1 if empty, 0 otherwise
int queue_full(Queue *q);               // returns 1 if full, 0 otherwise

#endif
```

Note: Students are expected to implement the queue.c file with the operations defined in queue.h.

input.txt

```
6
101 Alice 5
102 Bob 3
103 Charlie 7
104 Diana 4
105 Evan 6
106 Fiona 2
```

Expected output

```
Serving ticket 101 (Alice) at minute 0
Serving ticket 102 (Bob) at minute 5
Serving ticket 103 (Charlie) at minute 8
Serving ticket 104 (Diana) at minute 15
Serving ticket 105 (Evan) at minute 19
Serving ticket 106 (Fiona) at minute 25
```