--Double pointers--

1.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


struct item_details {

   int ID;

   char name[50];

   char category[30];

};


union attributes {

   float weight;

   float volume;

};


struct inventory {

   struct item_details item;

   union attributes attr;

};


void add_item(struct inventory **inventory, int *size, int id, const char *name, const char *category,
float value, int is_weight) {

   *size += 1;

   *inventory = realloc(*inventory, *size * sizeof(struct inventory));

   (*inventory)[*size - 1].item.ID = id;

   strcpy((*inventory)[*size - 1].item.name, name);

   strcpy((*inventory)[*size - 1].item.category, category);
```

```
    if (is_weight) {
        (*inventory)[*size - 1].attr.weight = value;
    } else {
        (*inventory)[*size - 1].attr.volume = value;
    }
}

void display_inventory(struct inventory *inventory, int size) {
    for (int i = 0; i < size; i++) {
        printf("ID: %d\nName: %s\nCategory: %s\n", inventory[i].item.ID, inventory[i].item.name,
inventory[i].item.category);
        printf("Weight: %.2f\n", inventory[i].attr.weight);  // Assuming weight is used
    }
}

int main() {
    struct inventory *inventory = NULL;
    int size = 0;

    add_item(&inventory, &size, 1, "Laptop", "Electronics", 1.5, 1);
    add_item(&inventory, &size, 2, "Table", "Furniture", 0.5, 0);
    display_inventory(inventory, size);

    free(inventory);
    return 0;
}
```

2.

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

```
#include <stdlib.h>

#include <string.h>


struct route_details {

   int ID;

   char start[30];

   char end[30];

};


union transport_modes {

   char mode[20];

};


struct route {

   struct route_details route_info;

   union transport_modes transport;

};


void add_route(struct route **routes, int *size, int id, const char *start, const char *end, const char
*mode) {

   *size += 1;

   *routes = realloc(*routes, *size * sizeof(struct route));

   (*routes)[*size - 1].route_info.ID = id;

   strcpy((*routes)[*size - 1].route_info.start, start);

   strcpy((*routes)[*size - 1].route_info.end, end);

   strcpy((*routes)[*size - 1].transport.mode, mode);

}


void display_routes(struct route *routes, int size) {
    for (int i = 0; i < size; i++) {
        printf("Route ID: %d\nFrom: %s\nTo: %s\nMode: %s\n", routes[i].route_info.ID,
routes[i].route_info.start, routes[i].route_info.end, routes[i].transport.mode);
    }
}

int main() {
    struct route *routes = NULL;
    int size = 0;

    add_route(&routes, &size, 1, "NY", "LA", "Truck");
    add_route(&routes, &size, 2, "SF", "Chicago", "Air");
    display_routes(routes, size);

    free(routes);
    return 0;
}
```

3.

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

struct vehicle_details {
    int ID;
    char type[30];
};

union vehicle_status {
    char status[20];
};

struct fleet {
    struct vehicle_details vehicle;
    union vehicle_status status;
};

void add_vehicle(struct fleet **fleet, int *size, int id, const char *type, const char *status) {
    *size += 1;
    *fleet = realloc(*fleet, *size * sizeof(struct fleet));
    (*fleet)[*size - 1].vehicle.ID = id;
    strcpy((*fleet)[*size - 1].vehicle.type, type);
    strcpy((*fleet)[*size - 1].status.status, status);
}

void display_fleet(struct fleet *fleet, int size) {
    for (int i = 0; i < size; i++) {
        printf("Vehicle ID: %d\nType: %s\nStatus: %s\n", fleet[i].vehicle.ID, fleet[i].vehicle.type,
fleet[i].status.status);
    }
}

int main() {
    struct fleet *fleet = NULL;
    int size = 0;

    add_vehicle(&fleet, &size, 1, "Truck", "Active");
    add_vehicle(&fleet, &size, 2, "Van", "Maintenance");
    display_fleet(fleet, size);

    free(fleet);
    return 0;
}
```

4.

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

struct order_details {
    int ID;
    char customer[50];
    int items;
};

union payment_methods {
    char method[20];
};

struct order {
    struct order_details order_info;
    union payment_methods payment;
};

void add_order(struct order **orders, int *size, int id, const char *customer, int items, const char *method) {
    *size += 1;
    *orders = realloc(*orders, *size * sizeof(struct order));
    (*orders)[*size - 1].order_info.ID = id;
    strcpy((*orders)[*size - 1].order_info.customer, customer);
    (*orders)[*size - 1].order_info.items = items;
    strcpy((*orders)[*size - 1].payment.method, method);
}

void display_orders(struct order *orders, int size) {
    for (int i = 0; i < size; i++) {
        printf("Order ID: %d\nCustomer: %s\nItems: %d\nPayment Method: %s\n", orders[i].order_info.ID,
orders[i].order_info.customer, orders[i].order_info.items, orders[i].payment.method);
    }
}

int main() {
    struct order *orders = NULL;
    int size = 0;

    add_order(&orders, &size, 1, "John", 3, "Credit Card");
    add_order(&orders, &size, 2, "Alice", 5, "Cash");
    display_orders(orders, size);

    free(orders);
    return 0;
}
```

5.

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

```
#include <string.h>

struct shipment_details {
    int tracking_number;
    char origin[50];
    char destination[50];
};

union tracking_events {
    char event[50];
};

struct shipment {
    struct shipment_details shipment_info;
    union tracking_events event;
};

void add_shipment(struct shipment **shipments, int *size, int tracking_number, const char *origin,
const char *destination, const char *event) {
    *size += 1;
    *shipments = realloc(*shipments, *size * sizeof(struct shipment));
    (*shipments)[*size - 1].shipment_info.tracking_number = tracking_number;
    strcpy((*shipments)[*size - 1].shipment_info.origin, origin);
    strcpy((*shipments)[*size - 1].shipment_info.destination, destination);
    strcpy((*shipments)[*size - 1].event.event, event);
}

void display_shipments(struct shipment *shipments, int size) {
    for (int i = 0; i < size; i++) {
        printf("Tracking Number: %d\nOrigin: %s\nDestination: %s\nEvent: %s\n",
shipments[i].shipment_info.tracking_number, shipments[i].shipment_info.origin,
shipments[i].shipment_info.destination, shipments[i].event.event);
    }
}

int main() {
    struct shipment *shipments = NULL;
    int size = 0;

    add_shipment(&shipments, &size, 1001, "NY", "LA", "Dispatched");
    add_shipment(&shipments, &size, 1002, "SF", "Chicago", "Delivered");
    display_shipments(shipments, size);

    free(shipments);
    return 0;
}
```

6.

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

struct traffic_node_details {
    int ID;
    char location[50];
};

union traffic_conditions {
    char condition[20];
};

struct traffic_data {
    struct traffic_node_details node_info;
    union traffic_conditions condition;
};

void add_traffic_node(struct traffic_data **traffic, int *size, int id, const char *location, const char
*condition) {
    *size += 1;
    *traffic = realloc(*traffic, *size * sizeof(struct traffic_data));
    (*traffic)[*size - 1].node_info.ID = id;
    strcpy((*traffic)[*size - 1].node_info.location, location);
    strcpy((*traffic)[*size - 1].condition.condition, condition);
}

void display_traffic(struct traffic_data *traffic, int size) {
    for (int i = 0; i < size; i++) {
        printf("Node ID: %d\nLocation: %s\nCondition: %s\n", traffic[i].node_info.ID,
traffic[i].node_info.location, traffic[i].condition.condition);
    }
}

int main() {
    struct traffic_data *traffic = NULL;
    int size = 0;

    add_traffic_node(&traffic, &size, 1, "NY - 5th Ave", "Congested");
    add_traffic_node(&traffic, &size, 2, "SF - Market St", "Clear");
    display_traffic(traffic, size);

    free(traffic);
    return 0;
}
```

7.

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

struct slot_details {
    int ID;
    char location[50];
    int size;
};

union item_types {
    char type[20];
};

struct warehouse_slot {
    struct slot_details slot_info;
    union item_types item;
};

void add_slot(struct warehouse_slot **slots, int *size, int id, const char *location, int size, const char
*item_type) {
    *size += 1;

    *slots = realloc(*slots, *size * sizeof(struct warehouse_slot));

    (*slots)[*size - 1].slot_info.ID = id;

    strcpy((*slots)[*size - 1].slot_info.location, location);

    (*slots)[*size - 1].slot_info.size = size;

    strcpy((*slots)[*size - 1].item.type, item_type);

}


void display_slots(struct warehouse_slot *slots, int size) {

    for (int i = 0; i < size; i++) {

        printf("Slot ID: %d\nLocation: %s\nSize: %d\nItem Type: %s\n", slots[i].slot_info.ID,
slots[i].slot_info.location, slots[i].slot_info.size, slots[i].item.type);

    }

}


int main() {

    struct warehouse_slot *slots = NULL;

    int size = 0;


    add_slot(&slots, &size, 1, "A1", 100, "Perishable");

    add_slot(&slots, &size, 2, "B2", 150, "Non-Perishable");

    display_slots(slots, size);


    free(slots);

    return 0;

}
```

8.

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

```
#include <stdlib.h>

#include <string.h>


struct package_details {

   int ID;

   float weight;

   char destination[50];

};


union delivery_methods {

   char method[20];

};


struct package {

   struct package_details package_info;

   union delivery_methods delivery;

};


void add_package(struct package **packages, int *size, int id, float weight, const char *destination,
const char *method) {

   *size += 1;

   *packages = realloc(*packages, *size * sizeof(struct package));

   (*packages)[*size - 1].package_info.ID = id;

   (*packages)[*size - 1].package_info.weight = weight;

   strcpy((*packages)[*size - 1].package_info.destination, destination);

   strcpy((*packages)[*size - 1].delivery.method, method);

}


void display_packages(struct package *packages, int size) {
   for (int i = 0; i < size; i++) {
       printf("Package ID: %d\nWeight: %.2f\nDestination: %s\nDelivery Method: %s\n", 
packages[i].package_info.ID, packages[i].package_info.weight, packages[i].package_info.destination, 
packages[i].delivery.method);
   }
}


int main() {
   struct package *packages = NULL;
   int size = 0;


   add_package(&packages, &size, 1, 5.0, "LA", "Express");
   add_package(&packages, &size, 2, 2.5, "NY", "Standard");
   display_packages(packages, size);


   free(packages);
   return 0;
}
```

9.

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


struct analytics_record {
    int ID;
    char timestamp[30];
};
```

```
union metrics {
    float speed;
    float efficiency;
};

struct analytics {
    struct analytics_record record_info;
    union metrics metric;
};

void add_analytics(struct analytics **data, int *size, int id, const char *timestamp, float value, int
is_speed) {
    *size += 1;
    *data = realloc(*data, *size * sizeof(struct analytics));
    (*data)[*size - 1].record_info.ID = id;
    strcpy((*data)[*size - 1].record_info.timestamp, timestamp);
    if (is_speed) {
        (*data)[*size - 1].metric.speed = value;
    } else {
        (*data)[*size - 1].metric.efficiency = value;
    }
}

void display_analytics(struct analytics *data, int size) {
    for (int i = 0; i < size; i++) {
        printf("ID: %d\nTimestamp: %s\nSpeed: %.2f\nEfficiency: %.2f\n", data[i].record_info.ID,
data[i].record_info.timestamp, data[i].metric.speed, data[i].metric.efficiency);
    }
}
```

```
int main() {
    struct analytics *data = NULL;
    int size = 0;

    add_analytics(&data, &size, 1, "2025-01-22 09:00", 60.5, 1);
    add_analytics(&data, &size, 2, "2025-01-22 10:00", 80.5, 0);
    display_analytics(data, size);

    free(data);
    return 0;
}
```

10.

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

struct schedule_details {
    int ID;
    char start_time[30];
    char end_time[30];
};

union transport_types {
    char type[20];
};

struct transportation_schedule {
    struct schedule_details schedule_info;

    union transport_types transport;

};


void add_schedule(struct transportation_schedule **schedule, int *size, int id, const char *start_time,
const char *end_time, const char *type) {

    *size += 1;

    *schedule = realloc(*schedule, *size * sizeof(struct transportation_schedule));

    (*schedule)[*size - 1].schedule_info.ID = id;

    strcpy((*schedule)[*size - 1].schedule_info.start_time, start_time);

    strcpy((*schedule)[*size - 1].schedule_info.end_time, end_time);

    strcpy((*schedule)[*size - 1].transport.type, type);

}


void display_schedule(struct transportation_schedule *schedule, int size) {

    for (int i = 0; i < size; i++) {

        printf("Schedule ID: %d\nStart Time: %s\nEnd Time: %s\nTransport Type: %s\n",
schedule[i].schedule_info.ID, schedule[i].schedule_info.start_time, schedule[i].schedule_info.end_time,
schedule[i].transport.type);

    }

}


int main() {

    struct transportation_schedule *schedule = NULL;

    int size = 0;


    add_schedule(&schedule, &size, 1, "9:00 AM", "12:00 PM", "Bus");

    add_schedule(&schedule, &size, 2, "1:00 PM", "4:00 PM", "Truck");

    display_schedule(schedule, size);
    free(schedule);

    return 0;

}
```

11.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


struct supplier_customer_details {

    int ID;

    char name[50];

};


union transaction_types {

    char transaction[20];

};


struct supply_chain {

    struct supplier_customer_details sc_info;

    union transaction_types transaction;

};


void add_transaction(struct supply_chain **chain, int *size, int id, const char *name, const char
*transaction_type) {

    *size += 1;

    *chain = realloc(*chain, *size * sizeof(struct supply_chain));

    (*chain)[*size - 1].sc_info.ID = id;

    strcpy((*chain)[*size - 1].sc_info.name, name);
    strcpy((*chain)[*size - 1].transaction.transaction, transaction_type);
}

void display_chain(struct supply_chain *chain, int size) {
    for (int i = 0; i < size; i++) {
        printf("ID: %d\nName: %s\nTransaction: %s\n", chain[i].sc_info.ID, chain[i].sc_info.name,
chain[i].transaction.transaction);
    }
}

int main() {
    struct supply_chain *chain = NULL;
    int size = 0;

    add_transaction(&chain, &size, 1, "Supplier 1", "Delivery");
    add_transaction(&chain, &size, 2, "Customer 1", "Purchase");
    display_chain(chain, size);

    free(chain);
    return 0;
}
```

12.

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

struct cargo_route_details {
    int ID;
    char start_location[50];

    char end_location[50];

};


union route_types {

    char route[20];

};


struct cargo_route {

    struct cargo_route_details route_info;

    union route_types type;

};


void add_route(struct cargo_route **routes, int *size, int id, const char *start_location, const char
*end_location, const char *route_type) {

    *size += 1;

    *routes = realloc(*routes, *size * sizeof(struct cargo_route));

    (*routes)[*size - 1].route_info.ID = id;

    strcpy((*routes)[*size - 1].route_info.start_location, start_location);

    strcpy((*routes)[*size - 1].route_info.end_location, end_location);

    strcpy((*routes)[*size - 1].type.route, route_type);

}


void display_routes(struct cargo_route *routes, int size) {

    for (int i = 0; i < size; i++) {

        printf("Route ID: %d\nStart: %s\nEnd: %s\nRoute Type: %s\n", routes[i].route_info.ID,
routes[i].route_info.start_location, routes[i].route_info.end_location, routes[i].type.route);

    }

}
```

```
int main() {
    struct cargo_route *routes = NULL;
    int size = 0;

    add_route(&routes, &size, 1, "NY", "LA", "Express");
    add_route(&routes, &size, 2, "SF", "Chicago", "Standard");
    display_routes(routes, size);

    free(routes);
    return 0;
}
```

13.

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

struct delivery_info {
    int ID;
    char delivery_time[30];
    float performance_score;
};

union performance_types {
    char status[20];
};

struct delivery_performance {
    struct delivery_info info;

    union performance_types status;

};


void add_performance(struct delivery_performance **performance, int *size, int id, const char
*delivery_time, float score, const char *status) {

    *size += 1;

    *performance = realloc(*performance, *size * sizeof(struct delivery_performance));

    (*performance)[*size - 1].info.ID = id;

    strcpy((*performance)[*size - 1].info.delivery_time, delivery_time);

    (*performance)[*size - 1].info.performance_score = score;

    strcpy((*performance)[*size - 1].status.status, status);

}


void display_performance(struct delivery_performance *performance, int size) {

    for (int i = 0; i < size; i++) {

        printf("ID: %d\nDelivery Time: %s\nPerformance Score: %.2f\nStatus: %s\n", performance[i].info.ID,
performance[i].info.delivery_time, performance[i].info.performance_score,
performance[i].status.status);

    }

}


int main() {

    struct delivery_performance *performance = NULL;

    int size = 0;


    add_performance(&performance, &size, 1, "2025-01-22 10:00", 95.0, "Good");

    add_performance(&performance, &size, 2, "2025-01-22 11:00", 85.0, "Average");

    display_performance(performance, size);
    free(performance);
    return 0;
}
```

14.

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

struct stock_details {
    int ID;
    char item[50];
    int quantity;
};

union stock_status {
    char status[20];
};

struct stock_replenishment {
    struct stock_details stock_info;
    union stock_status status;
};

void add_stock(struct stock_replenishment **stock, int *size, int id, const char *item, int quantity, const
char *status) {
    *size += 1;
    *stock = realloc(*stock, *size * sizeof(struct stock_replenishment));
    (*stock)[*size - 1].stock_info.ID = id;
    strcpy((*stock)[*size - 1].stock_info.item, item);
    (*stock)[*size - 1].stock_info.quantity = quantity;
    strcpy((*stock)[*size - 1].status.status, status);
}

void display_stock(struct stock_replenishment *stock, int size) {
    for (int i = 0; i < size; i++) {
        printf("Stock ID: %d\nItem: %s\nQuantity: %d\nStatus: %s\n", stock[i].stock_info.ID,
stock[i].stock_info.item, stock[i].stock_info.quantity, stock[i].status.status);
    }
}

int main() {
    struct stock_replenishment *stock = NULL;
    int size = 0;

    add_stock(&stock, &size, 1, "Item1", 100, "Replenished");
    add_stock(&stock, &size, 2, "Item2", 50, "Low");
    display_stock(stock, size);

    free(stock);
    return 0;
}
```

15.

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

```
struct delivery_point {

    int ID;

    char location[50];

};


union delivery_modes {

    char mode[20];

};


struct last_mile_delivery {

    struct delivery_point point_info;

    union delivery_modes mode;

};


void add_delivery_point(struct last_mile_delivery **delivery, int *size, int id, const char *location, const
char *mode) {

    *size += 1;

    *delivery = realloc(*delivery, *size * sizeof(struct last_mile_delivery));

    (*delivery)[*size - 1].point_info.ID = id;

    strcpy((*delivery)[*size - 1].point_info.location, location);

    strcpy((*delivery)[*size - 1].mode.mode, mode);

}


void display_delivery(struct last_mile_delivery *delivery, int size) {

    for (int i = 0; i < size; i++) {

        printf("Point ID: %d\nLocation: %s\nDelivery Mode: %s\n", delivery[i].point_info.ID,
delivery[i].point_info.location, delivery[i].mode.mode);

    }

}
```

```
int main() {
    struct last_mile_delivery *delivery = NULL;
    int size = 0;

    add_delivery_point(&delivery, &size, 1, "NYC", "Bike");
    add_delivery_point(&delivery, &size, 2, "SF", "Drone");
    display_delivery(delivery, size);

    free(delivery);
    return 0;
}
```

16.

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

struct robot_details {
    int ID;
    char type[30];
    char status[20];
};

union task_types {
    char picking[20];
    char sorting[20];
};
```

```
struct robot_coordination {

    struct robot_details robot_info;

    union task_types task;

};


void add_robot(struct robot_coordination **robots, int *size, int id, const char *type, const char *status,
const char *task_type) {

    *size += 1;

    *robots = realloc(*robots, *size * sizeof(struct robot_coordination));

    (*robots)[*size - 1].robot_info.ID = id;

    strcpy((*robots)[*size - 1].robot_info.type, type);

    strcpy((*robots)[*size - 1].robot_info.status, status);

    strcpy((*robots)[*size - 1].task.picking, task_type);

}


void display_robots(struct robot_coordination *robots, int size) {

    for (int i = 0; i < size; i++) {

        printf("Robot ID: %d\nType: %s\nStatus: %s\nTask: %s\n", robots[i].robot_info.ID,
robots[i].robot_info.type, robots[i].robot_info.status, robots[i].task.picking);

    }

}


int main() {

    struct robot_coordination *robots = NULL;

    int size = 0;


    add_robot(&robots, &size, 1, "Picker", "Idle", "Picking");

    add_robot(&robots, &size, 2, "Sorter", "Working", "Sorting");

    display_robots(robots, size);
    free(robots);

    return 0;

}
```

17.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


struct feedback_details {

    int ID;

    char content[100];

};


union feedback_types {

    char positive[10];

    char negative[10];

};


struct customer_feedback {

    struct feedback_details feedback_info;

    union feedback_types type;

};


void add_feedback(struct customer_feedback **feedback, int *size, int id, const char *content, const
char *type) {

    *size += 1;

    *feedback = realloc(*feedback, *size * sizeof(struct customer_feedback));
    (*feedback)[*size - 1].feedback_info.ID = id;
    strcpy((*feedback)[*size - 1].feedback_info.content, content);

    if (strcmp(type, "positive") == 0) {
        strcpy((*feedback)[*size - 1].type.positive, "Positive");
    } else {
        strcpy((*feedback)[*size - 1].type.negative, "Negative");
    }
}

void display_feedback(struct customer_feedback *feedback, int size) {
    for (int i = 0; i < size; i++) {
        printf("ID: %d\nContent: %s\nFeedback: %s\n", feedback[i].feedback_info.ID,
feedback[i].feedback_info.content, feedback[i].type.positive);
    }
}

int main() {
    struct customer_feedback *feedback = NULL;
    int size = 0;

    add_feedback(&feedback, &size, 1, "Great Service!", "positive");
    add_feedback(&feedback, &size, 2, "Late Delivery", "negative");
    display_feedback(feedback, size);

    free(feedback);
    return 0;
}
```

18.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


struct fleet_details {

   int ID;

   char location[50];

   char status[20];

};


union coordination_types {

   char dispatch[20];

   char reroute[20];

};


struct fleet_coordination {

   struct fleet_details fleet_info;

   union coordination_types coordination;

};


void add_fleet(struct fleet_coordination **fleets, int *size, int id, const char *location, const char
*status, const char *coordination_type) {

   *size += 1;

   *fleets = realloc(*fleets, *size * sizeof(struct fleet_coordination));

   (*fleets)[*size - 1].fleet_info.ID = id;

   strcpy((*fleets)[*size - 1].fleet_info.location, location);
```

```
    strcpy((*fleets)[*size - 1].fleet_info.status, status);
    strcpy((*fleets)[*size - 1].coordination.dispatch, coordination_type);
}

void display_fleets(struct fleet_coordination *fleets, int size) {
    for (int i = 0; i < size; i++) {
        printf("Fleet ID: %d\nLocation: %s\nStatus: %s\nCoordination: %s\n", fleets[i].fleet_info.ID,
fleets[i].fleet_info.location, fleets[i].fleet_info.status, fleets[i].coordination.dispatch);
    }
}

int main() {
    struct fleet_coordination *fleets = NULL;
    int size = 0;

    add_fleet(&fleets, &size, 1, "NYC", "Active", "Dispatch");
    add_fleet(&fleets, &size, 2, "SF", "Idle", "Reroute");
    display_fleets(fleets, size);

    free(fleets);
    return 0;
}
```

19.

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

struct security_event {
    int ID;

    char description[100];

};


union event_types {

    char breach[20];

    char resolved[20];

};


struct security_management {

    struct security_event event_info;

    union event_types event_type;

};


void add_event(struct security_management **events, int *size, int id, const char *description, const
char *event_type) {

    *size += 1;

    *events = realloc(*events, *size * sizeof(struct security_management));

    (*events)[*size - 1].event_info.ID = id;

    strcpy((*events)[*size - 1].event_info.description, description);


    if (strcmp(event_type, "breach") == 0) {

        strcpy((*events)[*size - 1].event_type.breach, "Breach");

    } else {

        strcpy((*events)[*size - 1].event_type.resolved, "Resolved");

    }

}


void display_events(struct security_management *events, int size) {
    for (int i = 0; i < size; i++) {
        printf("Event ID: %d\nDescription: %s\nEvent Type: %s\n", events[i].event_info.ID,
events[i].event_info.description, events[i].event_type.breach);
    }
}

int main() {
    struct security_management *events = NULL;
    int size = 0;

    add_event(&events, &size, 1, "Unauthorized Access", "breach");
    add_event(&events, &size, 2, "System Check", "resolved");
    display_events(events, size);

    free(events);
    return 0;
}
```

20.

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

struct billing_details {
    int ID;
    double amount;
    char date[20];
};
```

```
union payment_methods {

    char bank_transfer[20];

    char cash[10];

};


struct billing_system {

    struct billing_details bill_info;

    union payment_methods payment;

};


void add_bill(struct billing_system **bills, int *size, int id, double amount, const char *date, const char
*payment_method) {

    *size += 1;

    *bills = realloc(*bills, *size * sizeof(struct billing_system));

    (*bills)[*size - 1].bill_info.ID = id;

    (*bills)[*size - 1].bill_info.amount = amount;

    strcpy((*bills)[*size - 1].bill_info.date, date);


    if (strcmp(payment_method, "bank") == 0) {

        strcpy((*bills)[*size - 1].payment.bank_transfer, "Bank Transfer");

    } else {

        strcpy((*bills)[*size - 1].payment.cash, "Cash");

    }

}


void display_bills(struct billing_system *bills, int size) {

    for (int i = 0; i < size; i++) {
        printf("Bill ID: %d\nAmount: %.2f\nDate: %s\nPayment Method: %s\n", bills[i].bill_info.ID,
bills[i].bill_info.amount, bills[i].bill_info.date, bills[i].payment.bank_transfer);
    }
}

int main() {
    struct billing_system *bills = NULL;
    int size = 0;

    add_bill(&bills, &size, 1, 250.50, "2025-01-22", "bank");
    add_bill(&bills, &size, 2, 150.00, "2025-01-22", "cash");
    display_bills(bills, size);

    free(bills);
    return 0;
}
```