# Scenario 1:

1.With built-in function

#include <stdio.h>

#include <math.h> // For fmax (you can adapt if needed)

float calculate\_salary\_with\_builtin(float base\_salary, int experience\_years) {

float bonus\_percentage = fmin(experience\_years \* 0.10, 0.50); // Using fmin to cap at 50%

float bonus = base\_salary \* bonus\_percentage;

return base\_salary + bonus;

}

int main() {

float base;

int years;

printf("Enter base salary: ");

scanf("%f", &base);

printf("Enter years of experience: ");

scanf("%d", &years);

float total\_salary = calculate\_salary\_with\_builtin(base, years);

printf("Total salary (with built-in): %.2f\n", total\_salary);

return 0;

}

2.Without built-in function

#include <stdio.h>

float calculate\_salary\_without\_builtin(float base\_salary, int experience\_years) {

float bonus\_percentage = 0.0f;

if (experience\_years >= 1) {

if (experience\_years <= 5) {

bonus\_percentage = experience\_years \* 0.10f;

} else {

bonus\_percentage = 0.50f;

}

}

float bonus = base\_salary \* bonus\_percentage;

return base\_salary + bonus;

}

int main() {

float base;

int years;

printf("Enter base salary: ");

scanf("%f", &base);

printf("Enter years of experience: ");

scanf("%d", &years);

float total\_salary = calculate\_salary\_without\_builtin(base, years);

printf("Total salary (without built-in): %.2f\n", total\_salary);

return 0;

}

# Scenario 2:

1.With built-in function

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void formatUsernameBuiltin(char \*name) {

int i, j = 0;

for (i = 0; name[i] != '\0'; i++) {

if (name[i] != ' ') {

name[j++] = tolower(name[i]);

}

}

name[j] = '\0';

}

2.Without built-in function

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void formatUsernameManual(char \*name) {

int i, j = 0;

for (i = 0; name[i] != '\0'; i++) {

if (name[i] != ' ') {

if (name[i] >= 'A' && name[i] <= 'Z') {

name[j++] = name[i] - 'A' + 'a'; // Manual lowercase conversion

} else {

name[j++] = name[i];

}

}

}

name[j] = '\0'; // Null-terminate

}

int main() {

char name1[100] = " John Doe "; // Needs to be an array for modification

formatUsernameBuiltin(name1);

printf("Username (Built-in): %s\n", name1); // Output: johndoe

char name2[100] = " Jane Doe ";

formatUsernameManual(name2);

printf("Username (Manual): %s\n", name2); // Output: janedoe

    return 0;

# Scenario 3: Problem Statement 1

1.With built-in function.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct {

int id;

char name[50];

char description[100];

float price;

int quantity;

} Product;

int compare\_prices(const void \*a, const void \*b) {

float price\_a = ((Product \*)a)->price;

float price\_b = ((Product \*)b)->price;

if (price\_a < price\_b) return -1;

if (price\_a > price\_b) return 1;

return 0;

}

Product products\_builtin[num\_products];

memcpy(products\_builtin, products, sizeof(products)); // Create a copy

sort\_products\_builtin(products\_builtin, num\_products);

printf("Sorted using built-in function:\n");

for (int i = 0; i < num\_products; i++) {

printf("ID: %d, Name: %s, Price: %.2f\n", products\_builtin[i].id, products\_builtin[i].name, products\_builtin[i].price);

}

return 0;}

int main() {

Product products[] = {

{1, "Product A", "Description A", 10.50, 100},

{2, "Product B", "Description B", 25.00, 50},

{3, "Product C", "Description C", 15.75, 75},

{4, "Product D", "Description D", 5.00, 200},

{5, "Product E", "Description E", 30.20, 25}

};

int num\_products = sizeof(products) / sizeof(products[0]);

Product products\_builtin[num\_products];

memcpy(products\_builtin, products, sizeof(products)); // Create a copy

sort\_products\_builtin(products\_builtin, num\_products);

printf("Sorted using built-in function:\n");

for (int i = 0; i < num\_products; i++) {

printf("ID: %d, Name: %s, Price: %.2f\n", products\_builtin[i].id, products\_builtin[i].name, products\_builtin[i].price);

}

return 0;

}

2.Without built-in function

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct {

int id;

char name[50];

char description[100];

float price;

int quantity;

} Product;

void sort\_products\_bubble(Product products[], int num\_products) {

for (int i = 0; i < num\_products - 1; i++) {

for (int j = 0; j < num\_products - i - 1; j++) {

if (products[j].price > products[j + 1].price) {

Product temp = products[j];

products[j] = products[j + 1];

products[j + 1] = temp;

}

}

}

}

int main() {

Product products[] = {

{1, "Product A", "Description A", 10.50, 100},

{2, "Product B", "Description B", 25.00, 50},

{3, "Product C", "Description C", 15.75, 75},

{4, "Product D", "Description D", 5.00, 200},

{5, "Product E", "Description E", 30.20, 25}

};

int num\_products = sizeof(products) / sizeof(products[0]);

Product products\_bubble[num\_products];

memcpy(products\_bubble, products, sizeof(products));

sort\_products\_bubble(products\_bubble, num\_products);

printf("\nSorted using bubble sort:\n");

for (int i = 0; i < num\_products; i++) {

printf("ID: %d, Name: %s, Price: %.2f\n", products\_bubble[i].id, products\_bubble[i].name, products\_bubble[i].price);

}

    return 0;

}

# Problem Statement 2:

1.With built-in function

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct {

int id;

char name[50];

float price;

} Product;

typedef struct {

Product product;

int quantity;

} CartItem;

void add\_to\_cart(CartItem cart[], int \*cart\_size, Product product, int quantity) {

for (int i = 0; i < \*cart\_size; i++) {

if (cart[i].product.id == product.id) {

cart[i].quantity += quantity;

return;

}

}

if (\*cart\_size < 100) {

cart[\*cart\_size].product = product;

cart[\*cart\_size].quantity = quantity;

(\*cart\_size)++;

} else {

printf("Cart is full. Cannot add more items.\n");

}

}

void remove\_from\_cart(CartItem cart[], int \*cart\_size, int product\_id) {

for (int i = 0; i < \*cart\_size; i++) {

if (cart[i].product.id == product\_id)

{

for (int j = i; j < \*cart\_size - 1; j++) {

cart[j] = cart[j + 1];

}

(\*cart\_size)--;

return;

}

}

}

void update\_quantity(CartItem cart[], int cart\_size, int product\_id, int quantity) {

for (int i = 0; i < cart\_size; i++) {

if (cart[i].product.id == product\_id) {

cart[i].quantity = quantity;

return;

}

}

}

float calculate\_total(CartItem cart[], int cart\_size) {

float total = 0;

for (int i = 0; i < cart\_size; i++) {

total += cart[i].product.price \* cart[i].quantity;

}

return total;

}

int main() {

Product products[] = {

{1, "Laptop", 999.99},

{2, "Mouse", 29.99},

{3, "Keyboard", 79.99}

};

CartItem cart[100];

int cart\_size = 0;

add\_to\_cart(cart, &cart\_size, products[0], 1);

add\_to\_cart(cart, &cart\_size, products[1], 3);

add\_to\_cart(cart, &cart\_size, products[2], 1);

printf("Total cost: $%.2f\n", calculate\_total(cart, cart\_size));

remove\_from\_cart(cart, &cart\_size, 2);

update\_quantity(cart, cart\_size, 1, 2);

printf("Updated total cost: $%.2f\n", calculate\_total(cart, cart\_size));

return 0;

}

2.Without built-in function

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct {

int id;

char name[50];

float price;

} Product;

typedef struct {

Product product;

int quantity;

} CartItem;

void add\_to\_cart(CartItem cart[], int \*cart\_size, Product product, int quantity) {

for (int i = 0; i < \*cart\_size; i++) {

if (cart[i].product.id == product.id) {

cart[i].quantity += quantity;

return;

}

}

if (\*cart\_size < 100) {

cart[\*cart\_size].product = product;

cart[\*cart\_size].quantity = quantity;

(\*cart\_size)++;

} else {

printf("Cart is full. Cannot add more items.\n");

}

}

void remove\_from\_cart(CartItem cart[], int \*cart\_size, int product\_id) {

for (int i = 0; i < \*cart\_size; i++) {

if (cart[i].product.id == product\_id)

{

for (int j = i; j < \*cart\_size - 1; j++) {

cart[j] = cart[j + 1];

}

(\*cart\_size)--;

return;

}

}

}

void update\_quantity(CartItem cart[], int cart\_size, int product\_id, int quantity) {

for (int i = 0; i < cart\_size; i++) {

if (cart[i].product.id == product\_id) {

cart[i].quantity = quantity;

return;

}

}

}

float calculate\_total(CartItem cart[], int cart\_size) {

float total = 0;

for (int i = 0; i < cart\_size; i++) {

total += cart[i].product.price \* cart[i].quantity;

}

return total;

}

int main() {

Product products[] = {

{1, "Laptop", 999.99},

{2, "Mouse", 29.99},

{3, "Keyboard", 79.99}

};

CartItem cart[100];

int cart\_size = 0;

add\_to\_cart(cart, &cart\_size, products[0], 1);

add\_to\_cart(cart, &cart\_size, products[1], 3);

add\_to\_cart(cart, &cart\_size, products[2], 1);

printf("Total cost: $%.2f\n", calculate\_total(cart, cart\_size));

remove\_from\_cart(cart, &cart\_size, 2);

update\_quantity(cart, cart\_size, 1, 2);

printf("Updated total cost: $%.2f\n", calculate\_total(cart, cart\_size));

return 0;

}