Assignment 5 - Trees

# Name- Akshat Chandrapatle Div – CS – A Roll no. – 45

You have to maintain information for a shop owner. For each of the products sold in his/hers shop the following information is kept: a unique code, a name, a price, amount in stock, date received, expiration date. For keeping track of its stock, the clerk would use a computer program based on a search tree data structure.

Write a program to help this person, by implementing the following operations:   
• Insert an item with all its associated data.  
 • Find an item by its code, and support updating of the item found.  
 • List valid items in order of their names.

Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Product {

int code;

char name[50];

double price;

int stock;

char date\_received[11];

char expiration\_date[11];

struct Product\* left;

struct Product\* right;

};

int isValidDate(char\* date\_str) {

// Check if the date is "N/A"

// Check the length of the date string

if (strlen(date\_str) != 10) {

return 0;

}

// Check that the date string contains dashes at the right positions

if (date\_str[4] != '-' || date\_str[7] != '-') {

return 0;

}

// Check that the year, month, and day parts of the date string are numeric

int year, month, day;

if (sscanf(date\_str, "%4d-%2d-%2d", &year, &month, &day) != 3) {

return 0;

}

// Check that the month and day parts of the date string are within the correct ranges

if (month < 1 || month > 12 || day < 1 || day > 31) {

return 0;

}

if ((month == 4 || month == 6 || month == 9 || month == 11) && day == 31) {

return 0;

}

if (month == 2) {

if (day > 29) {

return 0;

}

if (day == 29 && (year % 4 != 0 || (year % 100 == 0 && year % 400 != 0))) {

return 0;

}

}

return 1; // The date is valid

}

struct Product\* newProduct(int code, char name[], double price, int stock, char date\_received[], char expiration\_date[]) {

struct Product\* new\_product = (struct Product\*) malloc(sizeof(struct Product));

new\_product->code = code;

strcpy(new\_product->name, name);

new\_product->price = price;

new\_product->stock = stock;

strcpy(new\_product->date\_received, date\_received);

strcpy(new\_product->expiration\_date, expiration\_date);

new\_product->left = NULL;

new\_product->right = NULL;

return new\_product;

}

struct Product\* insertProduct(struct Product\* root, int code, char name[], double price, int stock, char date\_received[], char expiration\_date[]) {

if (root == NULL) {

return newProduct(code, name, price, stock, date\_received, expiration\_date);

}

if (code < root->code) {

root->left = insertProduct(root->left, code, name, price, stock, date\_received, expiration\_date);

} else if (code > root->code) {

root->right = insertProduct(root->right, code, name, price, stock, date\_received, expiration\_date);

} else {

printf("Error: Product code already exists.\n");

}

return root;

}

struct Product\* findProduct(struct Product\* root, int code) {

if (root == NULL) {

return NULL;

}

if (root->code == code) {

return root;

} else if (code < root->code) {

return findProduct(root->left, code);

} else {

return findProduct(root->right, code);

}

}

void updateProduct(struct Product\* product, char name[], double price, int stock, char date\_received[], char expiration\_date[]) {

strcpy(product->name, name);

product->price = price;

product->stock = stock;

strcpy(product->date\_received, date\_received);

strcpy(product->expiration\_date, expiration\_date);

}

void inOrderTraversal(struct Product\* root) {

if (root != NULL) {

inOrderTraversal(root->left);

if (strcmp(root->expiration\_date, "N/A") != 0) {

printf("%d %s %.2f %d %s %s\n", root->code, root->name, root->price, root->stock, root->date\_received, root->expiration\_date);

}

inOrderTraversal(root->right);

}

}

int main() {

struct Product\* root = NULL;

int choice, code, stock;

double price;

char name[50], date\_received[11], expiration\_date[11];

struct Product\* product;

printf("\n1. Insert a new product\n");

printf("2. Find a product \n");

printf("3. Update a product \n");

printf("4. List products: \n");

printf("5. Exit\n");

while (1) {

printf("Enter your choice: ");

scanf("%d", &choice);

printf("\n");

switch (choice) {

case 1:

printf("Enter product code: ");

scanf("%d", &code);

printf("Enter product name: ");

scanf(" %[^\n]", name);

printf("Enter product price: ");

scanf("%lf", &price);

printf("Enter product stock: ");

scanf("%d", &stock);

printf("Enter date received (YYYY-MM-DD): ");

scanf("%s", date\_received);

if (!isValidDate(date\_received)) {

printf("Error: Invalid date received.\n");

break;

}

printf("Enter expiration date (YYYY-MM-DD or N/A): ");

scanf("%s", expiration\_date);

if (!isValidDate(expiration\_date) && strcmp(expiration\_date, "N/A") != 0) {

printf("Error: Invalid expiration date.\n");

break;

}

if (strcmp(expiration\_date, "N/A") != 0 && strcmp(date\_received, expiration\_date) >= 0) {

printf("Error: Expiration date must be after date received.\n");

break;

}

root = insertProduct(root, code, name, price, stock, date\_received, expiration\_date);

printf("Product added successfully.\n");

break;

case 2:

printf("Enter product code: ");

scanf("%d", &code);

product = findProduct(root, code);

if (product == NULL) {

printf("Product not found.\n");

break;

}

printf("%d %s %.2f %d %s %s\n", product->code, product->name, product->price, product->stock, product->date\_received, product->expiration\_date);

break;

case 3:

printf("Enter product code: ");

scanf("%d", &code);

product = findProduct(root, code);

if (product == NULL) {

printf("Product not found.\n");

break;

}

printf("Enter new product name: ");

scanf(" %[^\n]", name);

printf("Enter new product price: ");

scanf("%lf", &price);

printf("Enter new product stock: ");

scanf("%d", &stock);

printf("Enter new date received (YYYY-MM-DD): ");

scanf("%s", date\_received);

if (!isValidDate(date\_received)) {

printf("Error: Invalid date received.\n");

break;

}

printf("Enter new expiration date (YYYY-MM-DD or N/A): ");

scanf("%s", expiration\_date);

if (!isValidDate(expiration\_date) && strcmp(expiration\_date, "N/A") != 0) {

printf("Error: Invalid expiration date.\n");

break;

}

if (strcmp(expiration\_date, "N/A") != 0 && strcmp(date\_received, expiration\_date) >= 0) {

printf("Error: Expiration date must be after date received.\n");

break;

}

updateProduct(product, name, price, stock, date\_received, expiration\_date);

printf("Product updated successfully.\n");

break;

case 4:

printf("Code Name\t Price Stock Date\_Received Expiration\_Date\n");

inOrderTraversal(root);

break;

case 5:

printf("Exiting...\n");

exit(0);

default:

printf("Invalid choice.\n");

break;

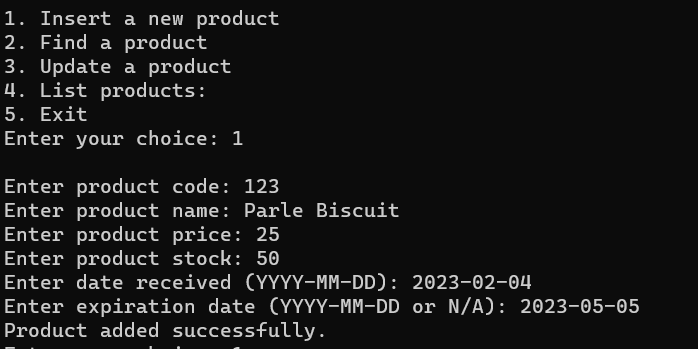
}

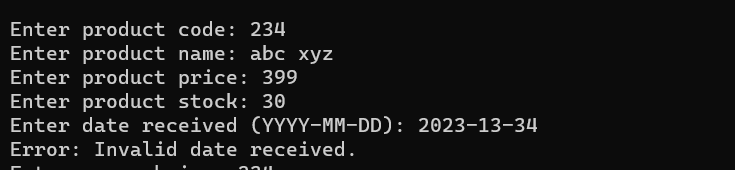
}

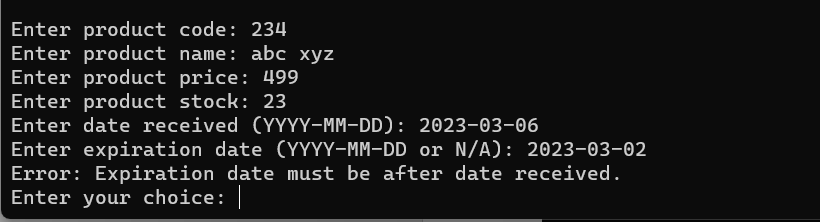
return 0;

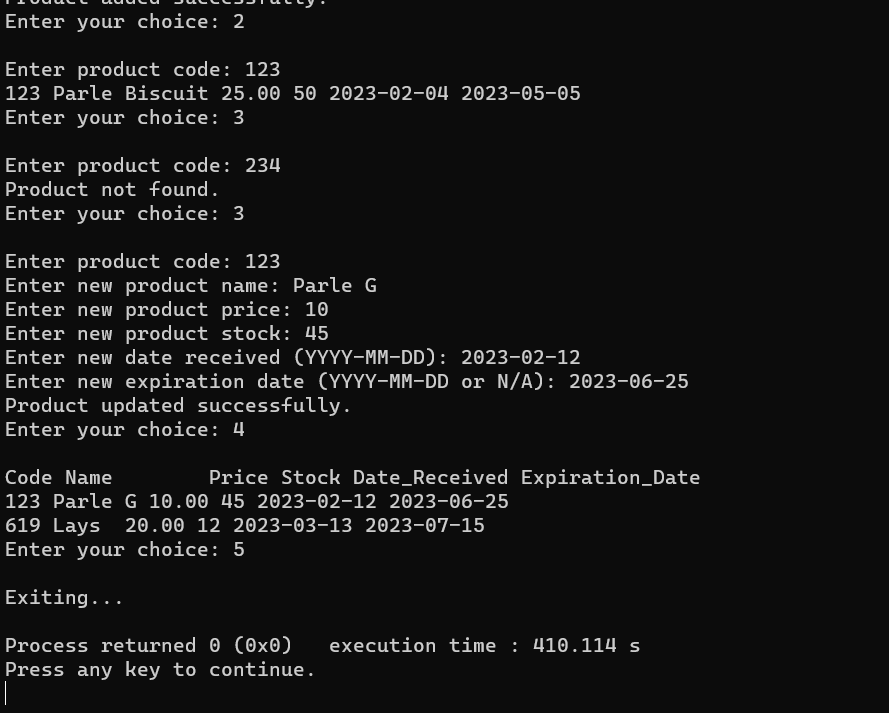
}

**OUTPUT:**

****

****

****

****