**Name: Khansa Afraz**

**SAP ID: 58935**

**Assignment (DSA theory)**

**Example 1: (Student Management System)**

#include <iostream>

using namespace std;

class Student {

public:

string name;

int rollNumber;

int marks[5];

float average=0;

int sum=0;

Student(string n, int r, int m[]) {

name = n;

rollNumber = r;

for (int i = 0; i < 5; i++) {

marks[i] = m[i];

}}

~Student() {

cout << "Deleting Student Record: " << name << endl;

}

void calculateAverage() {

for (int i = 0; i < 5; i++) {

sum += marks[i];

}

average = sum / 5.0;

cout << "Average Marks: " << average << endl;

}

void show() {

cout << "\n-----------------------------";

cout << "\nName: " << name;

cout << "\nRoll Number: " << rollNumber;

cout << "\nMarks: ";

for (int i = 0; i < 5; i++) {

cout << marks[i] << " ";

}

cout << "\n-----------------------------\n";

}

};

int main() {

int num;

cout << "Enter number of students: ";

cin >> num;

// Dynamically allocate an array of Student pointers

Student\* students[num];

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

string name;

int roll, marks[5];

cout << "\nEnter Name: ";

cin >> name;

cout << "Enter Roll Number: ";

cin >> roll;

cout << "Enter Marks (5 subjects): ";

for (int j = 0; j < 5; j++) {

cin >> marks[j];

}

students[i] = new Student(name, roll, marks);

}

cout << "\nStudent Details:\n";

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

students[i]->show();

students[i]->calculateAverage();

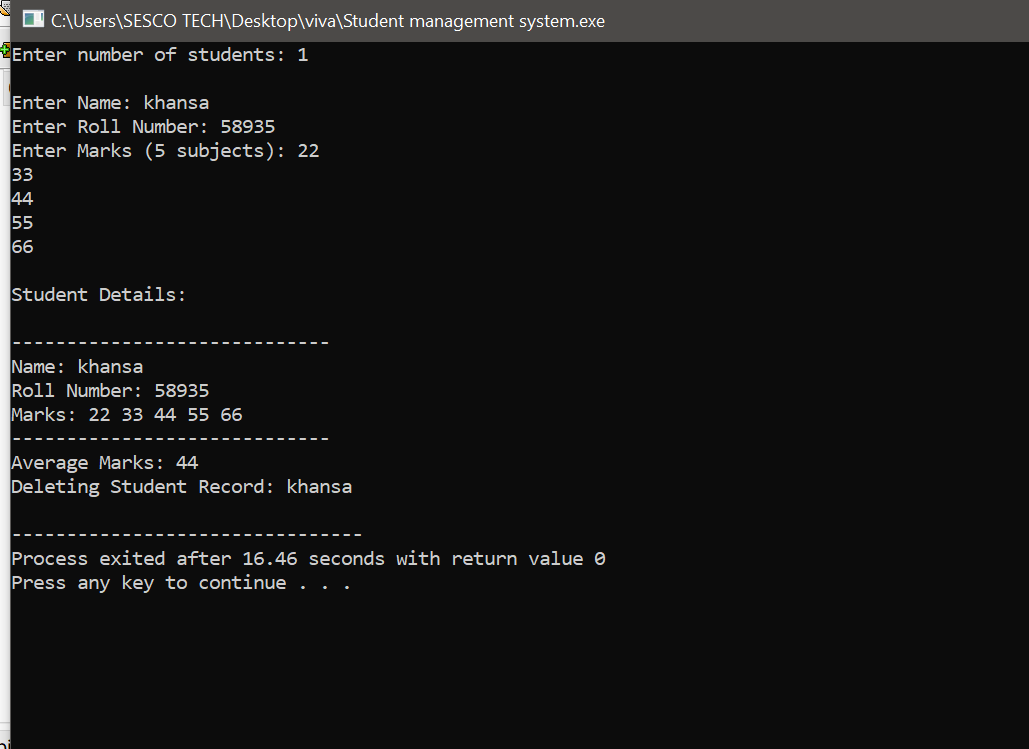
delete students[i];

}

return 0;

}

**Output:**



**Example 2:(Employee payroll system)**

#include <iostream>

using namespace std;

class Employee {

public:

string name;

int employeeID;

float basicSalary;

float allowances[3];

float totalEarnings;

Employee(string n, int id, float salary, float allow[]) {

name = n;

employeeID = id;

basicSalary = salary;

for (int i = 0; i < 3; i++) {

allowances[i] = allow[i];

}

totalEarnings = 0;

}

~Employee() {

cout << "Deleting Employee Record: " << name << endl;

}

void calculateTotalEarnings() {

float sum = 0;

for (int i = 0; i < 3; i++) {

sum = sum+allowances[i];

}

totalEarnings = basicSalary + sum;

cout << "Total Earnings: " << totalEarnings << endl;

}

void show() {

cout << "\n-----------------------------";

cout << "\nName: " << name;

cout << "\nEmployee ID: " << employeeID;

cout << "\nBasic Salary: " << basicSalary;

cout << "\nAllowances: ";

for (int i = 0; i < 3; i++) {

cout << allowances[i] << " ";

}

cout << "\n-----------------------------\n";

}

};

int main() {

int num;

cout << "Enter number of employees: ";

cin >> num;

Employee\* employees[num];

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

string name;

int id;

float salary, allowances[3];

cout << "\nEnter Name: ";

cin >> name;

cout << "Enter Employee ID: ";

cin >> id;

cout << "Enter Basic Salary: ";

cin >> salary;

cout << "Enter Allowances (3 quarterly values): ";

for (int i = 0; i < 3; i++) {

cin >> allowances[i];

}

employees[i] = new Employee(name, id, salary, allowances);

}

cout << "\nEmployee Details:\n";

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

employees[i]->show();

employees[i]->calculateTotalEarnings();

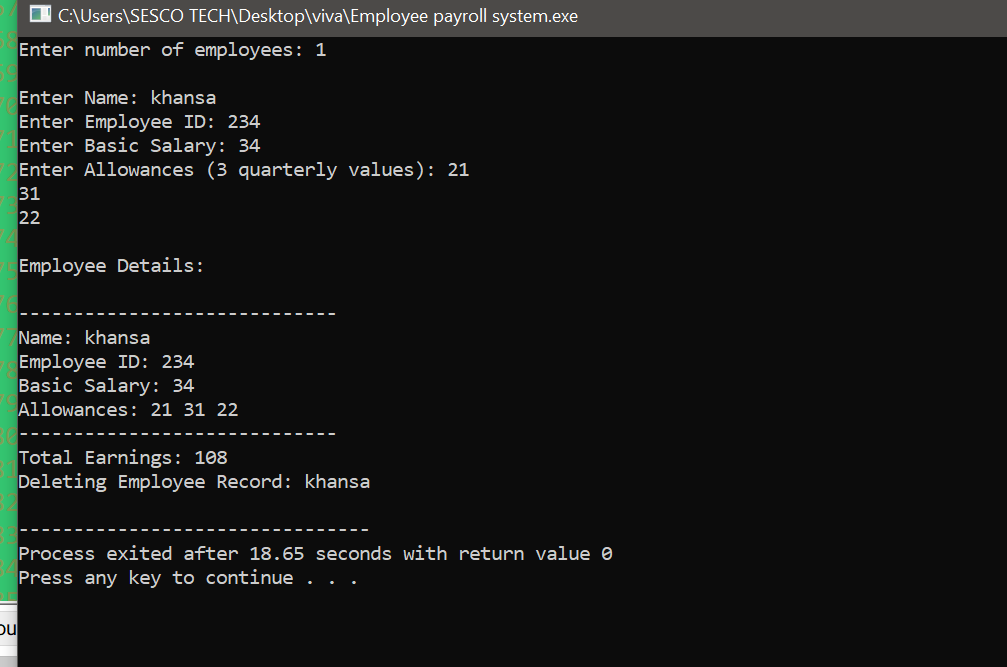
delete employees[i];

}

return 0;

}

**Output:**



**Example 3:(library book system)**

#include <iostream>

using namespace std;

class library {

public:

string title;

string author;

string ISBN;

int copies;

library(string t, string a, string i, int c) {

title = t;

author = a;

ISBN = i;

copies = c;

}

~library() {

cout << "Deleting book record: " << title << endl;

}

void issueBook() {

if (copies > 0) {

cout << "Book issued: " << title << endl;

copies--;

} else {

cout << "No copies available for: " << title << endl;

}

}

void returnBook() {

copies++;

cout << "Book returned: " << title << endl;

}

void display() {

cout << "\n-----------------------------";

cout << "\nTitle: " << title;

cout << "\nAuthor: " << author;

cout << "\nISBN: " << ISBN;

cout << "\nAvailable Copies: " << copies;

cout << "\n-----------------------------\n";

}

};

void searchBook(library\* books[], int num, string searchISBN) {

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

if (books[i]->ISBN == searchISBN) {

cout << "\nBook Found!";

books[i]->display();

return;

}

}

cout << "\nBook with ISBN " << searchISBN << " not found!\n";

}

int main() {

int num;

cout << "Enter number of books: ";

cin >> num;

library\* books[num];

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

string title, author, ISBN;

int copies;

cout << "\nEnter title: ";

cin >> title;

cout << "Enter author: ";

cin >> author;

cout << "Enter ISBN: ";

cin >> ISBN;

cout << "Enter number of copies: ";

cin >> copies;

books[i] = new library(title, author, ISBN, copies);

}

cout << "\nLibrary Details:\n";

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

books[i]->display();

books[i]->issueBook();

books[i]->returnBook();

books[i]->display();

}

string searchISBN;

cout << "\nEnter ISBN to search for a book: ";

cin >> searchISBN;

searchBook(books, num, searchISBN);

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

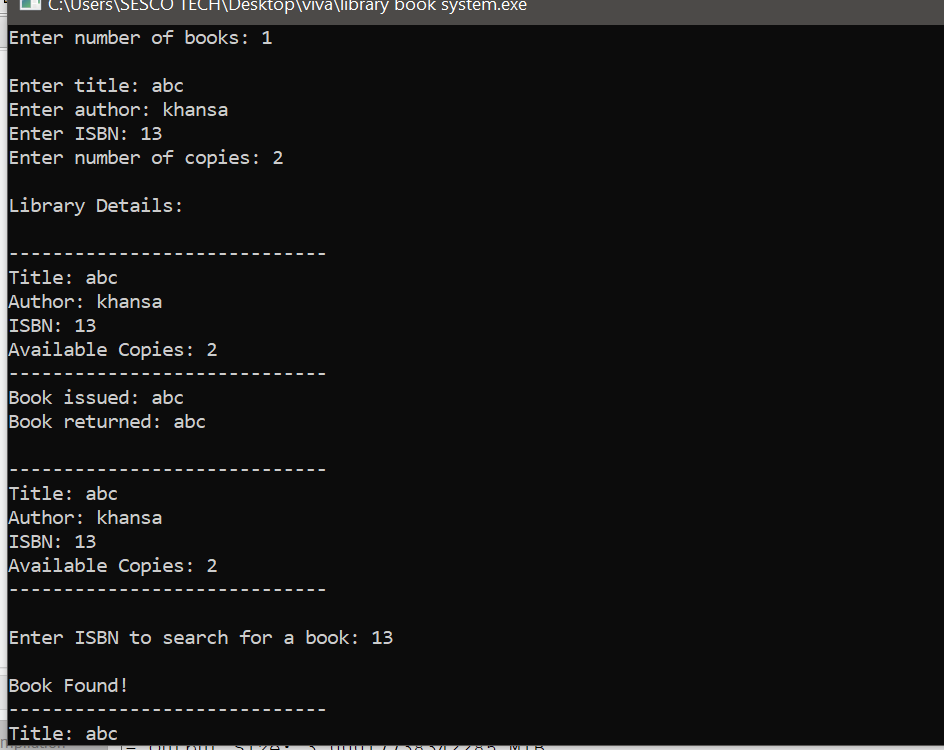
delete books[i];

}

return 0;

}

**Output:**



**Example 4(Product inventory management):**

#include <iostream>

using namespace std;

class Product {

public:

int productID;

string name;

float price;

int stockQuantity;

Product(int id, string n, float p, int stock) {

productID = id;

name = n;

price = p;

stockQuantity = stock;

}

~Product() {

cout << "Deleting Product: " << name << endl;

}

void sellProduct(int quantity) {

if (quantity > stockQuantity) {

cout << "Not enough stock available for " << name << "!" << endl;

} else {

stockQuantity -= quantity;

cout << "Sold " << quantity << " of " << name << ". Remaining stock: " << stockQuantity << endl;

if (stockQuantity < 5) {

cout << "Warning: Low stock for " << name << "!" << endl;

}

}

}

void show() {

cout << "\n-----------------------------";

cout << "\nProduct ID: " << productID;

cout << "\nName: " << name;

cout << "\nPrice: " << price;

cout << "\nStock Quantity: " << stockQuantity;

cout << "\n-----------------------------\n";

}

};

int main() {

int num;

cout << "Enter number of products: ";

cin >> num;

Product\* products[num];

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

int id, stock;

string name;

float price;

cout << "\nEnter Product ID: ";

cin >> id;

cout << "Enter Name: ";

cin >> name;

cout << "Enter Price: ";

cin >> price;

cout << "Enter Stock Quantity: ";

cin >> stock;

products[i] = new Product(id, name, price, stock);

}

cout << "\nProduct Inventory:\n";

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

products[i]->show();

}

int prodIndex, quantity;

cout << "\nEnter product index to sell (0 to " << num - 1 << "): ";

cin >> prodIndex;

cout << "Enter quantity to sell: ";

cin >> quantity;

if (prodIndex >= 0 && prodIndex < num) {

products[prodIndex]->sellProduct(quantity);

} else {

cout << "Invalid product index!" << endl;

}

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

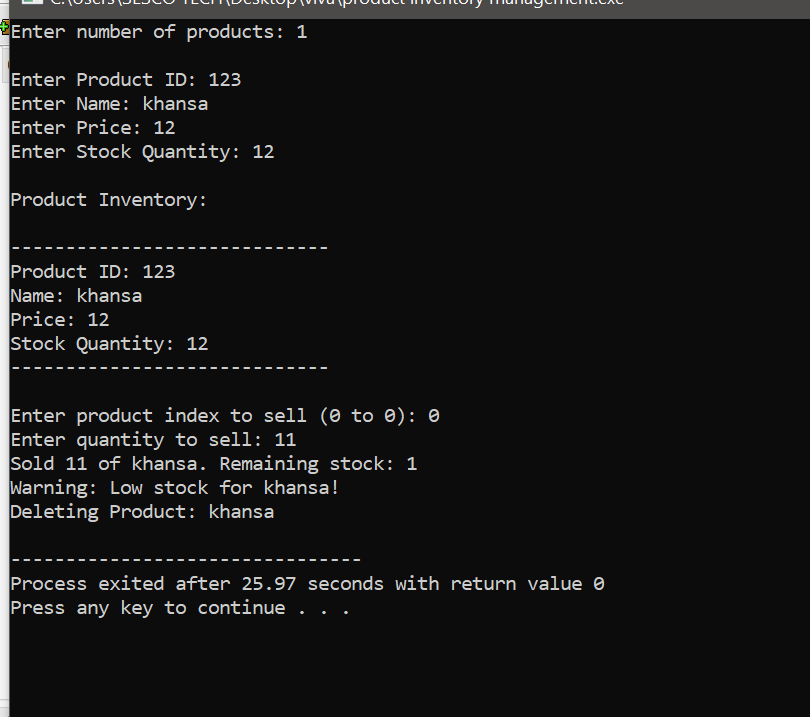
delete products[i];

}

return 0;

}

**Output:**



**Example 5:(Contact book)**

#include <iostream>

using namespace std;

class Contact {

public:

string name;

string phoneNumber;

Contact\* next;

Contact(string n, string p) {

name = n;

phoneNumber = p;

next = NULL;

}

~Contact() {

cout << "Deleting Contact: " << name << endl;

}

};

class ContactBook {

private:

Contact\* head;

public:

ContactBook() {

head = NULL;

}

~ContactBook() {

Contact\* current = head;

while (current != NULL) {

Contact\* temp = current;

current = current->next;

delete temp;

}

}

void addContact(string name, string phone) {

Contact\* newContact = new Contact(name, phone);

newContact->next = head;

head = newContact;

cout << "Contact added: " << name << endl;

}

void deleteContact(string name) {

Contact\* current = head;

Contact\* prev = NULL;

while (current != NULL && current->name != name) {

prev = current;

current = current->next;

}

if (current == NULL) {

cout << "Contact not found!" << endl;

return;

}

if (prev == NULL) {

head = current->next;

} else {

prev->next = current->next;

}

delete current;

cout << "Contact deleted: " << name << endl;

}

void displayContacts() {

Contact\* current = head;

if (!current) {

cout << "No contacts available." << endl;

return;

}

while (current != NULL) {

cout << current->name << " - " << current->phoneNumber << endl;

current = current->next;

}

}

void searchContact(string name) {

Contact\* current = head;

while (current != NULL) {

if (current->name == name) {

cout << "Found: " << current->name << " - " << current->phoneNumber << endl;

return;

}

current = current->next;

}

cout << "Contact not found!" << endl;

}

};

int main() {

ContactBook book;

int choice;

string name, phone;

while (true) {

cout << "\n1. Add Contact\n2. Delete Contact\n3. Display Contacts\n4. Search Contact\n5. Exit\nEnter your choice: ";

cin >> choice;

if (choice == 5) {

cout << "Exiting program." << endl;

break;

}

switch (choice) {

case 1:

cout << "Enter Name: ";

cin >> name;

cout << "Enter Phone Number: ";

cin >> phone;

book.addContact(name, phone);

break;

case 2:

cout << "Enter Name to delete: ";

cin >> name;

book.deleteContact(name);

break;

case 3:

book.displayContacts();

break;

case 4:

cout << "Enter Name to search: ";

cin >> name;

book.searchContact(name);

break;

default:

cout << "Invalid choice! Please try again." << endl;

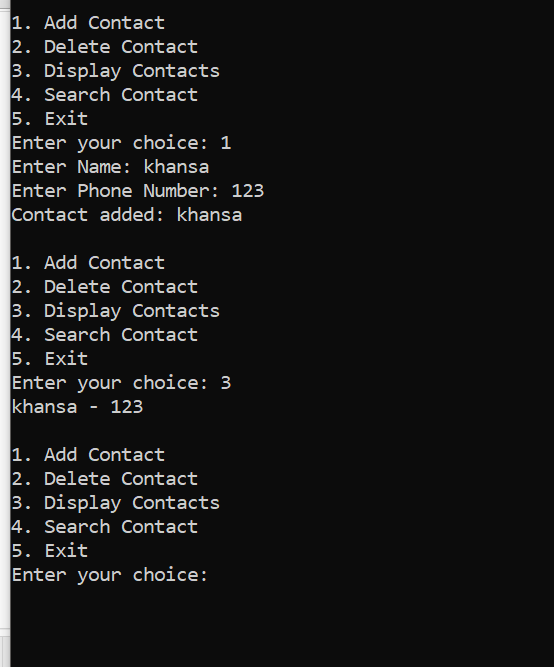
}

}

return 0;

}

**Output:**



**Example 6:(Ticket reservation system)**

#include <iostream>

using namespace std;

class Ticket {

public:

string passengerName;

int trainNumber;

int seatNumber;

Ticket(string name, int train, int seat) {

passengerName = name;

trainNumber = train;

seatNumber = seat;

}

};

class TicketReservation {

private:

Ticket\* seats[10];

public:

TicketReservation() {

for (int i = 0; i < 10; i++) {

seats[i] = NULL;

}

}

void reserveTicket(string name, int trainNumber) {

for (int i = 0; i < 10; i++) {

if (seats[i] == NULL) {

seats[i] = new Ticket(name, trainNumber, i + 1);

cout << "Ticket booked for " << name << " on Train " << trainNumber << " (Seat " << i + 1 << ")\n";

return;

}

}

cout << "No available seats!\n";

}

void cancelTicket(int seatNumber) {

if (seatNumber < 1 || seatNumber > 10 || seats[seatNumber - 1] == NULL) {

cout << "Invalid seat number!\n";

return;

}

delete seats[seatNumber - 1];

seats[seatNumber - 1] = NULL;

cout << "Seat " << seatNumber << " is now available.\n";

}

void displayTickets() {

bool found = false;

for (int i = 0; i < 10; i++) {

if (seats[i] != NULL) {

cout << "Seat " << seats[i]->seatNumber << ": " << seats[i]->passengerName << " (Train " << seats[i]->trainNumber << ")\n";

found = true;

}

}

if (!found) cout << "No tickets reserved.\n";

}

};

int main() {

TicketReservation system;

int choice, trainNumber, seatNumber;

string passengerName;

while (true) {

cout << "\n1. Reserve Ticket\n2. Cancel Ticket\n3. Display Reserved Tickets\n4. Exit\nEnter your choice: ";

cin >> choice;

if (choice == 1) {

cout << "Enter Passenger Name: ";

cin >> passengerName;

cout << "Enter Train Number: ";

cin >> trainNumber;

system.reserveTicket(passengerName, trainNumber);

} else if (choice == 2) {

cout << "Enter Seat Number to cancel: ";

cin >> seatNumber;

system.cancelTicket(seatNumber);

} else if (choice == 3) {

system.displayTickets();

} else if (choice == 4) {

break;

} else {

cout << "Invalid choice!\n";

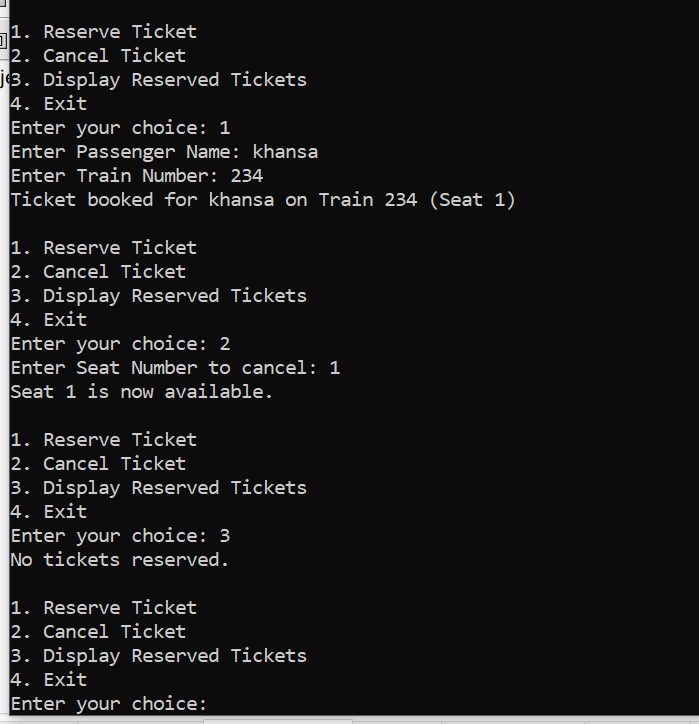
}

}

return 0;

}

**Output:**



**Example 7:(Vehicle parking system)**

#include <iostream>

using namespace std;

class Vehicle {

public:

string regNumber;

int slotNumber;

Vehicle(string reg, int slot) {

regNumber = reg;

slotNumber = slot;

}

~Vehicle() {

cout << "Vehicle with Reg No: " << regNumber << " removed from Slot: " << slotNumber << endl;

}

};

class ParkingLot {

private:

Vehicle\* slots[10];

int capacity;

public:

ParkingLot() {

capacity = 10;

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

slots[i] = NULL;

}

}

~ParkingLot() {

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

if (slots[i] != NULL) {

delete slots[i];

}

}

}

void addVehicle(string regNumber) {

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

if (slots[i] == NULL) {

slots[i] = new Vehicle(regNumber, i + 1);

cout << "Vehicle added at Slot: " << i + 1 << endl;

return;

}

}

cout << "No available slots!" << endl;

}

void removeVehicle(string regNumber) {

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

if (slots[i] != NULL && slots[i]->regNumber == regNumber) {

delete slots[i];

slots[i] = NULL;

cout << "Vehicle removed from Slot: " << i + 1 << endl;

return;

}

}

cout << "Vehicle not found!" << endl;

}

void displaySlots() {

cout << "Available slots: ";

bool found = false;

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

if (slots[i] == NULL) {

cout << (i + 1) << " ";

found = true;

}

}

if (!found) {

cout << "None";

}

cout << endl;

}

};

int main() {

ParkingLot lot;

int choice;

string regNumber;

while (true) {

cout << "\n1. Add Vehicle\n2. Remove Vehicle\n3. Display Available Slots\n4. Exit\nEnter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter Registration Number: ";

cin >> regNumber;

lot.addVehicle(regNumber);

break;

case 2:

cout << "Enter Registration Number to remove: ";

cin >> regNumber;

lot.removeVehicle(regNumber);

break;

case 3:

lot.displaySlots();

break;

case 4:

cout << "Exiting program." << endl;

break;

default:

cout << "Invalid choice! Please try again." << endl;

}

if (choice == 4) {

break;

}

}

return 0;

}

**Output:**

