CODE :-

#include<bits/stdc++.h>

#define max 12

Using namespace std;

Class vehicle{

Protected:

String number;

String type;

Public:

Vehicle(){}

Vehicle(string number,string type){

This->number=number;

This->type=type;

}

Virtual void display(){ //overriden function

Cout<<” Number: “<<number<<” Type: “<<type<<endl;

}

String getn(){

Return number;

}

};

Class bike:public vehicle{

Public:

Bike(string number){

This->number=number;

This->type=”Bike”;

}

Void display(){

Cout<<” BIKE: “<<endl;

Vehicle::display();

}

};

Class car:public vehicle{

Public:

Car(string number){

This->number=number;

This->type=”Car”;

}

Void display(){

Cout<<” CAR: “<<endl;

Vehicle::display();

}

};

Class other:public vehicle{

Public:

Other(string number){

This->number=number;

This->type=”Other vehicle”;

}

Void display(){

Cout<<” OTHER: “<<endl;

Vehicle::display();

}

};

Class parkingspot{

Vehicle \*obj;

Bool occupied;

Public:

Parkingspot(){

Obj=NULL;

Occupied=false;

}

Void park(vehicle \*v){

If(occupied){

Cout<<”Spot is already filled.”<<endl;

}

Else{

Obj=v;

Occupied=true;

Cout<<”Vehicle Parked Successfully.”<<endl;

}

}

Void remove(){

If(occupied){

Cout<<”Vehicle with number plate “<<obj->getn()<<” removed successfully.”<<endl;

Delete obj;

Obj=NULL;

Occupied=false;

}

Else{

Cout<<”Parking Slot is already empty.”<<endl;

}

}

Void display(){

If(occupied){

Obj->display();

}

Else{

Cout<<” Empty Spot”<<endl;

}

}

Bool isoccupied(){

Return occupied;

}

};

Int main(){

Parkingspot arr[max];

String number;

Int choice,slot;

While(1){

Cout<<”\*\*\*MENU\*\*\*”<<endl;

Cout<<”1.Park a bike”<<endl;

Cout<<”2.Park a car”<<endl;

Cout<<”3.Park any other vehicle”<<endl;

Cout<<”4.Remove a vehicle”<<endl;

Cout<<”5.Display all spots”<<endl;

Cout<<”6.Display empty spots”<<endl;

Cout<<”7.Exit”<<endl;

Cout<<”Enter your choice: “;

Cin>>choice;

Switch(choice){

Case 1:

Cout<<”Enter bike number : “;

Cin>>number;

Cout<<”Enter a slot for parking from 1 to “<<max<<” : “;

Cin>>slot;

If(slot>=1&&slot<=max){

Arr[slot-1].park(new bike(number));

}

Else{

Cout<<”Invalid spot number.”<<endl;

}

Break;

Case 2:

Cout<<”Enter car number : “;

Cin>>number;

Cout<<”Enter a slot for parking from 1 to “<<max<<” : “;

Cin>>slot;

If(slot>=1&&slot<=max){

Arr[slot-1].park(new car(number));

}

Else{

Cout<<”Invalid spot number.”<<endl;

}

Break;

Case 3:

Cout<<”Enter the vehicle number : “;

Cin>>number;

Cout<<”Enter a slot for parking from 1 to “<<max<<” : “;

Cin>>slot;

If(slot>=1&&slot<=max){

Arr[slot-1].park(new other(number));

}

Else{

Cout<<”Invalid spot number.”<<endl;

}

Break;

Case 4:

Cout<<”Enter a slot number to remove a vehicle from parking (1 to “<<max<<”): “;

Cin>>slot;

If(slot>=1&&slot<=max)

Arr[slot-1].remove();

Else

Cout<<”Invalid spot number.”<<endl;

Break;

Case 5:

Cout<<”All spots: “<<endl;

For(int i=0;i<max;i++){

Cout<<”Spot : “<<(i+1);

Arr[i].display();

}

Break;

Case 6:

Cout<<”Displaying empty spots: “<<endl;

For(int i=0;i<max;i++){

If(!arr[i].isoccupied()){

Cout<<”Spot: “<<(i+1)<<” : Empty”<<endl;

}

}

Break;

Case 7:

Cout<<”Exiting the program...”<<endl;

Exit(0);

Break;

Default:

Cout<<”Enter a correct choice.”<<endl;

}

}

Return 0;

}