```cpp
//
// Created by amirp on 12/9/2021.
//

#include "iostream"
#include "math.h"
#include "string"

using namespace std;

//create stack
template<typename T>
class Stack{

    T *arr;
    int nextIndex,capacity;
public:
    Stack(){
        capacity = 10;
        arr = new T[capacity];
        nextIndex = 0;
    }


    int size(){
        return nextIndex;
    }
    bool isEmpty()
    {
        if(nextIndex==0){
            return true;
        } else return false;
    }
    void push(T ele){
        T element=ele;
        if(nextIndex==capacity){
            T *newArr = new T[2*capacity];
            for (int i = 0; i < capacity; i++) {
                newArr[i]=arr[i];

            }
            delete []arr;
        }

        arr[nextIndex]=element;
        nextIndex++;
        capacity=2*capacity;

```

```
49      }
50
51      void pop(){
52          if(isEmpty()) {
53              cout << "Stack is Empty" << endl;
54              delete []arr;
55              return;
56          }
57          nextIndex--;
58
59      }
60
61      T top(){
62          if (isEmpty()){
63              cout<<"Stack empty"<<endl;
64              return 0;
65          }
66          return arr[nextIndex-1];
67      }
68
69      void checkArrayValue(){
70          for(int i=0;i<6;i++){
71              cout<<arr[i]<<endl;
72          }
73      }
74
75  };
76
77  //function for showing main menu and message
78
79  void showMenu(){
80      cout<<endl<<"Enter A for Arrival"<<endl;
81      cout<<"Enter D for Departure "<<endl;
82      cout<<"Enter S for Show All Cars"<<endl;
83      cout<<"Enter Input: ";
84  }
85  void printParkingMessage(){
86      cout<<endl<<"Your Car is park"<<endl;
87  }
88  void printDepatureMessage(){
89      cout<<"Your Car is Depature"<<endl;
90  }
91  void lisenceNumberInput(int *lisenceNumber){
92      cout<<"Enter Your Lisence Number";
93      cin>>*lisenceNumber;
94  }
95  void printparkingFullMessage(){
96      cout<<"Parking Full you cannot park right now sorry"<<endl;
```

```
 97 }
 98 void printNotParkingMessage(){
 99     cout<<"Your Car is Not parked Here!"<<endl;
100 }
101
102
103
104
105
106
107 class ParkingGarage:public Stack<int>{
108     int lisenceNumberPlate,carEntries[10],carEntriesIndex=0,
    car[10],arrival[10],depature[10],countArrival;
109     char userInput;
110
111
112     struct car{
113         int count=0;
114         int *numberPlate;
115     } carDetail[10];
116
117
118 //store lisenceNumber plate of Entered car in a array
119     void carEntry(){
120         carEntries[carEntriesIndex]=lisenceNumberPlate;
121         cout<<"The car slot: "<<carEntriesIndex+1;
122         carEntriesIndex++;
123
124     }
125     //find length of array
126     int findLength(int array,int baseArray){
127         cout<<array<<endl<<baseArray;
128         return array/ baseArray;
129     }
130     //check car is parked or not
131     bool isCarParked(){
132         int length = sizeof(carEntries)/sizeof(carEntries[0]);
133         for (int i=0;i<length;i++){
134             if(carEntries[i]==lisenceNumberPlate){
135                 return true;
136             }
137
138
139         }
140         return false;
141     }
142     //take out all car and push back after remove one car
143     void takeOutCar(int size,int index=0){
```

```
144         int sizeofStack = size;
145         if(isEmpty()|| index==size){
146             return;
147         }
148 
149         int holdStackData=top();
150         pop();
151 
152         takeOutCar(sizeofStack,index+1);
153         movedCount(index);
154 
155         if(index!=lisenceNumberPlate){
156             push(holdStackData);
157 
158 
159         }
160 
161     }
162     //after car depature remove car entry from an array
163     void removeFromCarEntry(){
164         int length= findLength(sizeof(carEntries), sizeof(
    carEntries[0]));
165         cout<<length;
166         int i;
167         for(i=0;i<10;i++){
168             if(carEntries[i]==lisenceNumberPlate){
169                 break;
170             }
171         }
172 
173         if(i<length)
174         {
175             length=length-1;
176             for(int j=i;j<length;j++){
177                 carEntries[j]=carEntries[j+1];
178 
179                 carEntriesIndex--;
180             }
181         }
182 
183         carEntriesIndex--;
184 
185         //countArrival--;
186     }
187     //function for take input from user
188     void takeChoices(){
189         showMenu();
190         cin>>userInput;
```

```
191
192     }
193     //total car
194     void totalCar(){
195         int car=size();
196         int length = sizeof(carEntries)/sizeof(carEntries[0]);
197         cout<<endl<<"total car Number : "<<car<<endl;
198
199
200     }
201     //display all cars
202     void showAllCars(){
203         int car=size();
204         for(int i=0;i<car;i++){
205             cout<<"["<<carEntries[i]<<"]"<<" , ";
206         }
207     }
208     void setArrival(){
209
210 //       int i=0;
211 //
212 //       arrival[lisenceNumberPlate][i]=size();
213 //       cout<<endl<<arrival[lisenceNumberPlate][i];
214 //       i++;
215 //
216 //        for(countArrival=0;countArrival>10;countArrival++){
217 //            carDetail[countArrival].numberPlate=&
    lisenceNumberPlate;
218 //        }
219         int countArrival=0;
220         arrival[carEntriesIndex]=countArrival;
221
222
223
224     }
225     void setDepature(){
226         int i=0;
227         depature;
228     }
229     void movedCount(int index){
230         int count=0;
231         arrival[index]=count;
232         count++;
233
234     }
235     void showTotalMoved(){
236         cout<<"Total Moved"<<" : "<<arrival[carEntriesIndex]<<
    endl;
```

```
237     }
238 
239 
240 public:
241     void carParkingOrDepature() {
242         while(true) {
243             takeChoices();
244 
245             if (userInput == 'a') {
246                 if (size() <= 12) {
247 
248                     lisenceNumberInput(&lisenceNumberPlate);
249                     push(lisenceNumberPlate);
250                     carEntry();
251                     setArrival();
252 
253                     printParkingMessage();
254 
255                 } else {
256                     printparkingFullMessage();
257                 }
258 
259             } else if (userInput == 'd') {
260                 if (!isEmpty()) {
261                     lisenceNumberInput(&lisenceNumberPlate);
262                     cout << lisenceNumberPlate;
263                     if(lisenceNumberPlate==top()){
264                         pop();
265                         printDepatureMessage();
266                         showTotalMoved();
267                         carEntriesIndex--;
268                     } else{
269                         if(isCarParked()){
270                             takeOutCar(size());
271                             showTotalMoved();
272                             removeFromCarEntry();
273                             printDepatureMessage();
274 
275                         } else{
276                             printNotParkingMessage();
277                         }
278                     }
279 
280 
281                 }
282                 else {
283                     printNotParkingMessage();
284                 }
```

```
285             }else if(userInput=='s'){
286                 showAllCars();
287             }
288             else{
289                 cout<<"Please Enter Correct Input";
290             }
291             totalCar();
292       }
293         }
294 
295 };
296 
297 //some function for infix expression and postfixexpression
298 
299 //check precendece
300 int precendence(char c) {
301     if (c == '^') {
302         return 3;
303     } else if (c == '*' || c == '/') {
304         return 2;
305     } else if (c == '+' || c == '-') {
306         return 1;
307     } else {
308         return -1;
309     }
310 }
311 
312 // check operands
313 
314 bool isOperand(char exp){
315     if(exp >='a'&& exp<='z'){
316             return true;
317     }
318     return false;
319 }
320 
321 
322 //Creating class for infix to postfix or prefix and evaluate
    them
323 
324 class InfixExpression:public Stack<char> {
325 
326 
327 
328 public:
329 
330     //Creating function to check balanced parenthesis
331 //
```

```
332 //    string isParenthesisBalanced(string infix){
333 //
334 //    }
335
336
337     //creating function to convert infix to Postfix
338     string infixToPostfix(string infix) {
339         string result;
340
341         for (int i = 0; i < infix.length(); i++) {
342             if (isOperand(infix[i])) {
343                 result += infix[i];
344
345             } else if (infix[i] == '(') {
346                 push(infix[i]);
347             } else if (infix[i] == ')') {
348                 while (!isEmpty() && top() != '(') {
349                     result += top();
350                     pop();
351
352                 }
353                 if (!isEmpty()) {
354                     pop();
355                 }
356
357             } else {
358                 while (!isEmpty() && precendence(top()) >=
    precendence(infix[i])) {
359                     result += top();
360                     pop();
361                 }
362                 push(infix[i]);
363             }
364
365         }
366         while (!isEmpty()) {
367             result += top();
368             pop();
369         }
370
371         return result;
372     }
373
374 };
375
376 //convert postfix expression to infix expression
377 class PostFixToInfix:public Stack<string>{
378     string operand,operand1,operand2;
```

```
379 public:
380     void postFixToInfix(string postfix) {
381         string result;
382         for (int i = 0; i < postfix.length(); i++) {
383             if (isOperand(postfix[i])) {
384                 operand=postfix[i];
385                 push(operand);
386 
387             } else {
388                 operand1 = top();
389                 pop();
390                 operand2 = top();
391                 pop();
392                 result = "(" + operand2 + postfix[i] +
    operand1 + ")";
393                 push(result);
394                 }
395             }
396 
397         cout<<top();
398 
399 
400         }
401 
402     };
403 
404 //};
405 int main()
406 {
407     ParkingGarage parking;
408     parking.carParkingOrDepature();
409 
410     InfixExpression infix;
411     string posfixExpression=infix.infixToPostfix("a+(((b-c)*(d
    -e)+f)/g)^(h-j)");
412     cout<<posfixExpression<<endl;
413     PostFixToInfix postfix;
414     postfix.postFixToInfix(posfixExpression);
415 
416     return 0;
417 
418 }
```