Question 1

Q.1
code:

```
class Solution{
public:
itn removeDuplicates(vector<int>&nums){
int j=1;
for(int i=1;i<nums.size();i++){
if(nums[i]!=nums[i-1]){
nums[j]=nums[i];
j++;
}
}
return j;
}
};
```

Question 2

Q.2

code: class Solution{

public:

bool isValidsudoku(vector<vector<char>> &board){

unordered_map<char,int>eachbox;

unordered_map<char,int>row;

unordered_map<char,int>column;

int j=0;

int i=0;

for(int i=0;i<9;i++){

for(int j=0;j<9;j++){

if(board[i][j]!='.'){

row[board[i][j]++;

}

if(board[i][j]!='.'){

coloumn[board[i][j]++;

}

if(row[board[i][j]]>1)

return false;

if(column[board[i][j]]>1)

return false;

}

row.clear();

column.clear();

}

for(int i=0;i<9;i+=3){

for(int j=0;j<9;j+=3){

eachbox.clear();

```
for(it x=i;x<i+3;x++){
for(it y=i;y<i+3;y++){
if(board[x][y]!='.'){
eachbox[board[x][y]]++;}
if(eachbox[board[x][y]>1)
return false;
 }
}
}
}
return true;
}
};
```

Question 3

Q.3

Code:

```
Class Solution{
public:
int maxProfit(vector<int>& prices){
int main_price=prices[0];
int maxprof=0;
for(int i =1i<prices.size();i++){
maxprof= max(maxprof,prices[i]-min_price);
min_price=min(prices[i],min_price);
}
return maxprof;
```

```
}
}
```

Question 4

Q.4

code:

```
 class Solution{
public:
int searchInsert(vector<int>& nums,int target){
int low=0;
int high=nums.size();
int mid;
if(target>nums[high-1]{
return high;
}
while(low<=high){
mid=(low+high)/2;
if(nums[mid]==target){
return mid;
}
if(target<numz[mid]){
high=mid-1;
}
else{
low=mid+1;
}
}
return low;
}
};
```

Question 5

Q.5

Code: class Solution {

public:

int cancompletecircuit(vector<int>& gas,vector<int>&cost){

int n =gas.size();

int total_gas=0,total_cost=0;

int curr_gas=0,starting_point=0;

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

total_gas+=gas[i];

total_cost+=cost[i];

curr_gas+=gas[i]-cost[i];

if(curr_gas<0){

starting_point=i+1;

curr_gas=0;

}

}

return(total_gas<total_cost)?-1:starting_point;

}

};

Question 6

Q.6

```
class Solution{

public:

int bottomUp(vector<int>&ums,int n ){

vector<int>dp(n+1,0);

dp[0]=nums[0];

for(int i =1;i<=n;i++){

int temp=0;

if(i-2>=0){

temp=dp[i-2];

int include=temp+nums[i];

int exclude=dp[i-1];

dp[i]=max(include,exclude);

}

return dp[n];

}

int rob(vector<int>&nums){

int n=nums.size()-1;

vector<int>dp(n+1,-1);

return bottonup(nums,n);

}

};
```