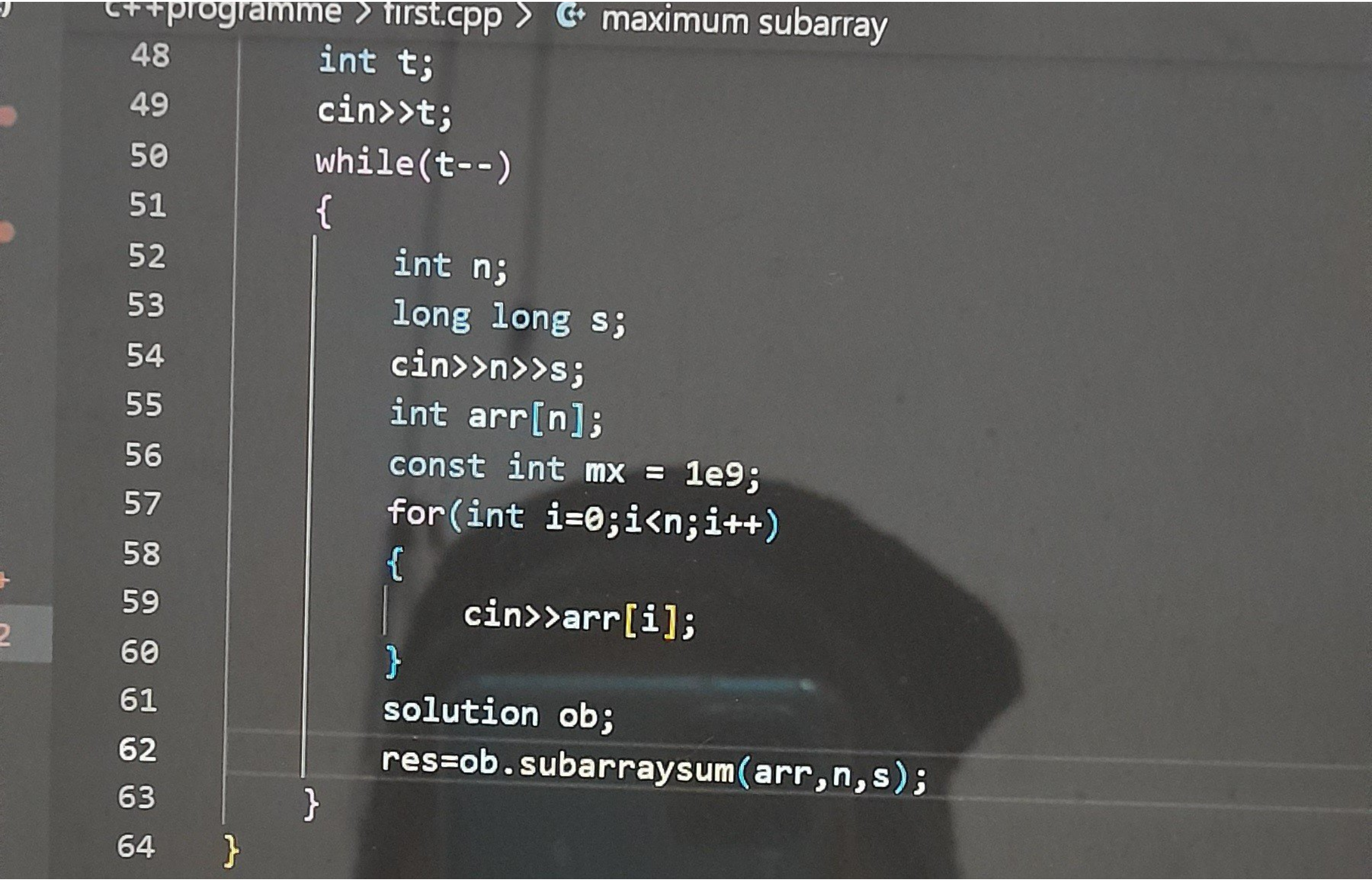
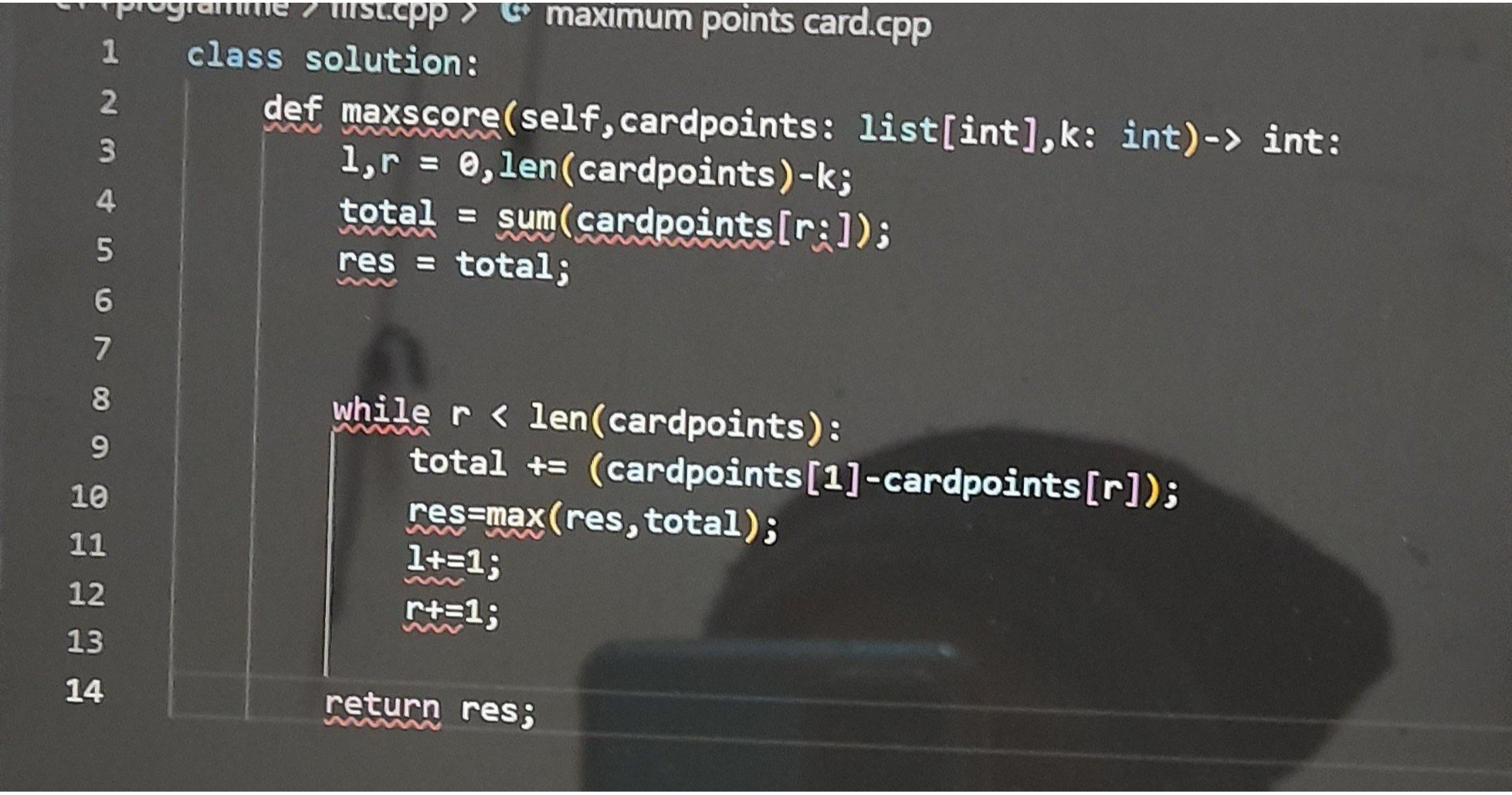
```cpp
#include<bits/stdc++.h>
using namespace std;


//} driver code ends
class solution
{
    public:
    //function to find a continuous sub-array
    vector<int> subarraysum(int arr[], int n,int i,int j)
    {
        //your code here
        int sas=0,i=0,c=0;
    vector<int> v;
            for(int j=0; j<n; j++)
            {
                if(sas+arr[j]<=s)
                {
                    sas+=arr[j];
                }
                else
                {
                    sas+=aar[j];
                    while(sas>s)
                    {
                        sas-=arr[i];
                        i++;
                    }
                }
```

```
27                                  i++;
28                              }
29                          }
30                          if(sas==s)
31                          {
32                              v.push_back(i+1);
33                              v.push_back(j+1);
34                              return v;
35                          }
36                      }
37                      if(c==0)
38                      {
39                          v.push_back(-1);
40                          return v;
41                      }
42              }
43      };
44      // { driver code starts.
45
46      int main()
47      {
48          int t;
49          cin>>t;
50          while(t--)
51          {
52              int n;
53              long long s;
54              cin>>n>>s;
55              int arr[n];
```


c++programme > first.cpp > maximum subarray
int t;
cin>>t;
while(t--)
{
int n;
long long s;
cin>>n>>s;
int arr[n];
const int mx = 1e9;
for(int i=0;i<n;i++)
{
cin>>arr[i];
}
solution ob;
res=ob.subarraysum(arr,n,s);
}
}


maximum points card.cpp
class solution:
def maxscore(self,cardpoints: list[int],k: int)-> int:
l,r = 0,len(cardpoints)-k;
total = sum(cardpoints[r:]);
res = total;
while r < len(cardpoints):
total += (cardpoints[l]-cardpoints[r]);
res=max(res,total);
l+=1;
r+=1;
return res;