به نام خدا

برنامه نویسی پیشرفته(AP) **امیرحسین زمانی-۹۴۲۳۸۰۲**
گزارش تمرین سری چهارم **۱۳۹۸/۰۲/۱۸**

لینک این سری از تمرین بر روی **github** :

https://github.com/AHHHZ975/AP-HW4

سوال ۱ :

a) یک راه بهینه برای انتقال منابع حافظه از یک object به object دیگر و بدون استفاده از copy شدن است. در واقع به عبارت بهتر با استفاده از move semantic می توان حتی به یک متغیر temporary آدرس داد و آن را به صورت pass by value به یک تابع فرستاد.

b) به معنای چندریختی است. درواقع این چند ریختی در class ها زمانی اتفاق می افتد که یک سلسه مراتبی مثلا به صورت ارث بری والد و فرزند وجود داشته باشد و بخواهیم به یک member function از کلاس والد اشاره کنیم که در کلاس فرزند هم وجود دارد. بدین صورت یک تابع به صورت ها و ریخت های مختلف وجود دارد که این مفهوم را می رساند.

c) این مفهوم زمانی معنا پیدا می کند که نیاز باشد تا صرفا یک معماری و اسکلتی از یک کلاس ارایه شود. به عبارت دقیق تر یک کلاس که pure abstract است صرفا دارای prototype توابعی است که کلاس هایی که از آن به ارث می برند دارای difinition دقیق آن ها هستند و درواقع موجودیت و اسکلت و بدنه خود را مشخص می کند و دارای هیچگونه متغیر یا تعریف دقیق تابعی نیست. به طور کلی از pure abstrac class ها برای نوشتن interface ها به کاربرده می شود. زیرا صرفا نیاز است تا رابطی برای محیط بیرون از کد و برنامه برقرار شود.

d) برای بازتعریفی دوباره یک member function در کلاس فرزند یا همان derived class که ابتدا در کلاس base یا والد تعریف شده است استفاده می شود.

e) با نوشتن inline پشت توابع می توان به compiler فهماند که به جای اینکه درهنگام اجرای این برنامه، در run time به این تابع ارجاع داده شود، در compile time آن را جایگزین کند و از آن به بعد با آن به صورت یک macro که از قبل تعریف شده است رفتار می کند و با اینکار زمان اجرای برنامه کاهش می یابد معمولا برای توابعی که از نظر دستورالعملی و کاری که انجام می دهند، بزرگ هستند از inline استفاده می شود.

f) دربرخی از مواقع که یک کلاس، constructor هایی با یک پارامتر ورودی دارد، برای این که compiler با استفاده از پارامتر ورودی در تشخیص استفاده از constructor اشتباه نشود(implicit conversion را انجام ندهد) با گذاشتن لغت explicit پشت نام constructor از این اشتباه جلوگیری می شود.

سوال ۲ :

مقدار تابع size چه زمانی که از تابع reserve استفاده شود و چه استفاده نشود، همواره برابر مقداری است که وکتور دارای المان های با مقدار است.
اما اگر از تابع reserve استفاده شود، تابع capacity نزدیکترین توان عدد ۲ را به ظرفیت مورد نیاز حافظه برای وکتور اختصاص می دهد. یعنی در این مثال اگر ورودی ۱۰۰۰ به تابع reserve داده شود تابع capacity نزدیک ترین توان عدد ۲ یعنی ۱۰۲۴ را برمیگرداند. این بدین علت است که از همان اول برای حافظه مشخص شده است که وکتور چقدر از حافظه را اشغال خواهد کرد و درنتیجه مقدار تابع capacity مقداری ثابت است.
اما اگر از تابع reserve استفاده نشود، تابع capacity درهرمرحله نزدیکترین توان عدد ۲ که بزرگتر از مقدار تابع size است را برمی گرداند و این بدین منزله است که تا زمانی که مقدار size برابر با capacity نشود مقداری که از حافظه برای آن وکتور گرفته می شود، تغییری نخواهد کرد. یعنی اگر مثلا در حال حاضر ۱۰ مقدار داخل وکتور ریخته شده است، مقدار size برابر ۱۰ و مقدار capacity برابر ۱۶ یعنی ۲ به توان ۴ است. فکر می کنم این موضوع به این دلیل است که تخصیص حافظه به صورت بهینه انجام شود.

```
11
12  template <typename dataType>
13  void getSizeAndCapacityOfvector(vector<dataType>* aVector){
14      cout << "The size is: " << aVector->size() << endl;
15      cout << "The capacity is: " << aVector->capacity() << endl;
16  }
17
18  int main(){
19      vector<std::unique_ptr<string>> aVector;
20      for(size_t i {0}; i != 1000; i++){
21          aVector.push_back(std::make_unique<string>("Str" + std::to_string(i)));
22          getSizeAndCapacityOfvector(&aVector);
23      }
24      for(size_t i {0}; i != aVector.size(); i++){
25          cout << *aVector.at(i) << endl;
26      }
27      ////////////////////////////////////////////////////////////
28      aVector.clear();
29      aVector.reserve(1000);
30      for(size_t i {0}; i != 1000; i++){
31          aVector.push_back(std::make_unique<string>("Str" + std::to_string(i)));
32          getSizeAndCapacityOfvector(&aVector);
33      }
34      for(size_t i {0}; i != 1000; i++){
35          cout << *aVector.at(i) << endl;
36      }
37      return 0;
38  }
```

سوال ۳ :

دقیقا عین خواسته سوال کد پیاده سازی و نتایج زیر حاصل شد: (البته خیلی تمیزتر و اصولی تر میشد که کلاس ها و توابع پیاده سازی شوند، اما وقت نکردم! :))

```
ahz@AHZ:~/Desktop/AP-HW4-9423802/Q3$ ./main
Circle radius = 3.5
center->(6, 9)
area of 38.4845

Square side length = 12
center->(2, 2)
area of 144

Sphere radius = 5
center->(1.5, 4.5, 0)
area of 314.159 & volume of 523.599

Cube side length = 2.2
center->(0, 0, 0)
area of 29.04 & volume of 10.648
```

سوال ۴:

در این سوال با توجه به main داده شده حتما باید توابع volume و area و print به صورت virtual تعریف شوند. زیرا هرموقع که یک object از کلاس والد به صورت پوینتری ساخته می شود تمام توابع مشترک بین والد و فرزند، به صورت توابع فرزند فراخوانی می شوند. در اینجا هم چون در تابع main داده شده در سوال shape ها را به صورت پوینتری ساخته باید توابع virtual باشند تا هنگامی که در حلقه for قرار است مقادیر موردنظر نشان داده شوند، توابع مخصوص به هر فرزند فراخوانی شود، در غیر این صورت ممکن است مثلا تابع volume برای یک مربع فراخوانی شود که بی معنی است!
به طورکلی هر زمان که نیاز باشد تا فقط توابع فرزند فراخوانی شود(حتی زمانی که یک object از کلاس والد ساخته شده است) نیاز است تا آن تابع به صورت virtual تعریف شود.

سوال ۵:

دراین سوال مشابه روند موجود در کتاب، یعنی به صورت Nested class عمل شد و نتایج زیر حاصل شد.

```
ahz@AHZ:~/Desktop/AP-HW4-9423802/Q5$ ./main
TEXTZ
TEXTY
TEXTX
TEXTW
TEXTV
TEXTU
TEXTT
TEXTS
TEXTR
TEXTQ
TEXTP
TEXTO
TEXTN
TEXTM
TEXTL
TEXTK
TEXTJ
TEXTI
TEXTH
TEXTG
TEXTF
TEXTE
TEXTD
TEXTC
TEXTB
TEXTA
Stack is empty
ahz@AHZ:~/Desktop/AP-HW4-9423802/Q5$
```

سوال ۶:

الف) با استفاده از دستور std::remove نمی توان دقیقا خود وکتور موردنظر را از المانی مشخص خالی کرد. زیرا این دستور، خود دارای یک خروجی از جنس vecotr<int>::iterator است که آرایه مطلوب را در آن می ریزد. حال اگر یک وکتور از این جنس تعریف کنیم و خروجی دستور remove را در آن بریزیم، وکتور مطلوب را خواهیم داشت. به صورت زیر:

```
vector<int> vec{1, 2, 3, 4, 5, 4, 3, 2, 1};
////////////////// Print original vector ////////////
cout << "Original vector : ";
for(size_t i{}; i != vec.size(); i++){
    cout << " " << vec.at(i);
}
cout << endl;
///////////////// After remove //////////////////
vector<int>::iterator afterRemove;
afterRemove = std::remove(vec.begin(), vec.end(),2);
cout << "After remove : ";
for (auto p = vec.begin(); p != afterRemove; p++){
    cout << " " << *p;
}
cout << endl;
```

```
ahz@AHZ:~/Desktop/AP-HW4-9423802/Q6$ ./main
Original vector :  1 2 3 4 5 4 3 2 1
After remove :  1 3 4 5 4 3 1
```

حال اگر بخواهیم مقدار ۲ را دقیقا از وکتور اولیه حذف کنیم از ترکیب دستور erase و remove به صورت زیر استفاده می کنیم:

```
30      ////////////////////////////////////////////////////
31      vec.erase(std::remove(vec.begin(), vec.end(), 2), vec.end());
32      cout << "After remove : ";
33      for(size_t i{}; i != vec.size(); i++){
34          cout << " " << vec.at(i);
35      }
36      cout << endl;
```

و داریم:

```
ahz@AHZ:~/Desktop/AP-HW4-9423802/Q6$ ./main
Original vector :  1 2 3 4 5 4 3 2 1
After remove :  1 3 4 5 4 3 1
```

ب) برای اینکار دو راه داریم:

اول استفاده از دستور std::for_each به صورت زیر:

```
37      ////////////////////////////////////////////////
38      std::for_each(vec.begin(), vec.end(), [](int& d) { d*=2.0;});
39      cout << "Duplicated vector: ";
40      std::ostream_iterator<int> out_it (std::cout," ");
41      std::copy (vec.begin(), vec.end(), out_it );
42      cout << endl;
```

و راه دوم استفاده از دستور std::tranform به صورت زیر:

```
////////////////////////////////////////////////
transform(vec.begin(), vec.end(), vec.begin(),std::bind2nd(std::multiplies<int>(), 2));
std::ostream_iterator<int> out_it (std::cout," ");
std::copy (vec.begin(), vec.end(), out_it );
cout << endl;
```

که درهر دو حالت نتیجه به شکل زیر است:

```
ahz@AHZ:~/Desktop/AP-HW4-9423802/Q6$ ./main
Original vector :  1 2 3 4 5 4 3 2 1
Duplicated vector: 2 4 6 8 10 8 6 4 2
```

برای نشان دادن وکتور بدون حلقه نیز از دستور copy و ostream_iterator به صورتی که در بالا نشان داده شده است، استفاده شده است.

ج)

با استفاده از دستور Accumulate و transform به صورت زیر:

```
51      ////////////////////////////
52      vector<double> tempVec{1, 2, 3, 4, 5, 4, 3, 2, 1};
53      double average = std::accumulate( vec.begin(), vec.end(), 0.0)/vec.size();
54      transform(tempVec.begin(), tempVec.end(), tempVec.begin(), std::bind2nd(std::plus<double>(), -average));
55      sort(tempVec.begin(), tempVec.end());
56      transform(tempVec.begin(), tempVec.end(), tempVec.begin(),std::bind2nd(std::plus<double>(), +average));
57      cout << "The sorted vector: ";
58      std::ostream_iterator<double> out_it (std::cout," ");
59      std::copy (tempVec.begin(), tempVec.end(), out_it );
60      cout << endl;
```

داریم:

```
/ahz@AHZ:~/Desktop/AP-HW4-9423802/Q6$ ./main
Original vector :  1 2 3 4 5 4 3 2 1
The sorted vector: 1 1 2 2 3 3 4 4 5
```

د) با استفاده از دستور unique مقادیر تکراری را پیدا کرده و با استفاده از دستور erase آن ها را حذف میکنیم.

```
    ////////////////////////////////////////////////////////
    sort(vec.begin(), vec.end());
    vec.erase(unique(vec.begin(), vec.end()), vec.end() );
    std::ostream_iterator<int> out_it (std::cout," ");
    cout << "The Non-repetitive vector: ";
    std::copy (vec.begin(), vec.end(), out_it );
    cout << endl;
```

```
ahz@AHZ:~/Desktop/AP-HW4-9423802/Q6$ ./main
Original vector :  1 2 3 4 5 4 3 2 1
The Non-repetitive vector: 1 2 3 4 5
```

ر)

```
68      ////////////////////////////////////////////////////////
69      std::set<double> s(vec.begin(), vec.end());
70      std::set<double>::iterator it;
71      it = s.lower_bound(4);
72      s.erase(it, s.end());
73      std::ostream_iterator<int> out_it (std::cout," ");
74      std::copy (s.begin(), s.end(), out_it );
75      cout << endl;
```

```
ahz@AHZ:~/Desktop/AP-HW4-9423802/Q6$ ./main
Original vector :  1 2 3 4 5 4 3 2 1
Desired vector: 1 2 3
```

سوال ۷:

برای این سوال از list استفاده شد.

برای ریختن آرایه درون list:

```
16  int main(){
17      srand(time(NULL));
18      list<int> aList;
19      list<int> b(50);
20      list<int> c(50);
21      int a[50]{};
22      std::ostream_iterator<int> out_it (std::cout," ");
23      vector<int>::iterator it;
24      ////////////////////////////////////////////////////////
25      aList.assign(a, a + 50);
26      cout << "a is: " << endl;
27      std::copy(aList.begin(), aList.end(), out_it);
28      cout << endl;
```

برای تولید اعداد غیرتکرای و تصادفی درون آرایه:

```
29      ////////////////////////////////////////////////////////
30      std::generate(b.begin(), b.end(), generateRandomNumber);
31      cout << "The random generated array is: " << endl;
32      std::copy(b.begin(), b.end(), out_it);
33      cout << endl;
34      b.sort();
35      b.unique();
36      cout << "The unique random generated array is: " << endl;
37      std::copy(b.begin(), b.end(), out_it);
38      cout << endl;
```

```
The random generated array is:
21 0 16 8 1 45 48 19 39 39 15 43 41 17 41 48 1 4 14 4 8 49 39 4 49 46 40 35 35 32 6 8 32 23 18 33 20 17 4 12 8 20 5 2 37 48 0 41 4 16
The unique random generated array is:
0 1 2 4 5 6 8 12 14 15 16 17 18 19 20 21 23 32 33 35 37 39 40 41 43 45 46 48 49
```

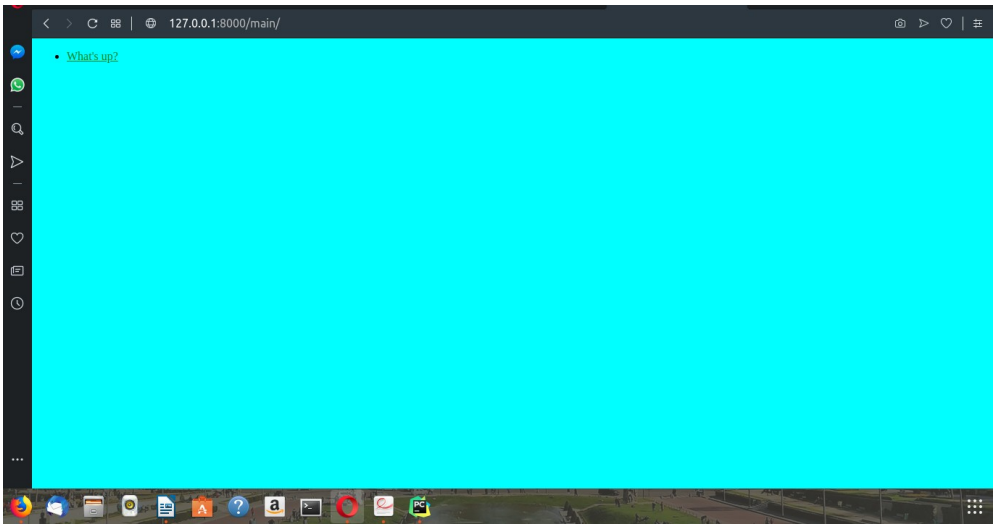
برای تفاضل مربعات دو list:

```
39      ////////////////////////////////////////////////////////
40      std::generate(c.begin(), c.end(), generateRandomNumber);
41      cout << "Another random generated array is: " << endl;
42      std::copy(c.begin(), c.end(), out_it);
43      cout << endl;
44      c.sort();
45      c.unique();
46      cout << "The unique random generated array is: " << endl;
47      std::copy(c.begin(), c.end(), out_it);
48      cout << endl;
49      transform(b.begin(), b.end(), b.begin(), b.begin(), std::multiplies<int>());
50      transform(c.begin(), c.end(), c.begin(), c.begin(), std::multiplies<int>());
51      if(c.size() < b.size()){
52          c.insert(c.end(), b.size() - c.size(), 0);
53      }
54      else if(b.size() < c.size()){
55          b.insert(b.end(), c.size() - b.size(), 0);
56      }
57      int results[c.size()];
58      transform(c.begin(), c.end(), b.begin(), results, std::minus<int>());
59      cout << "The minus result is: " << endl;
60      std::copy(results, results + c.size(), out_it);
61      cout << endl;
```

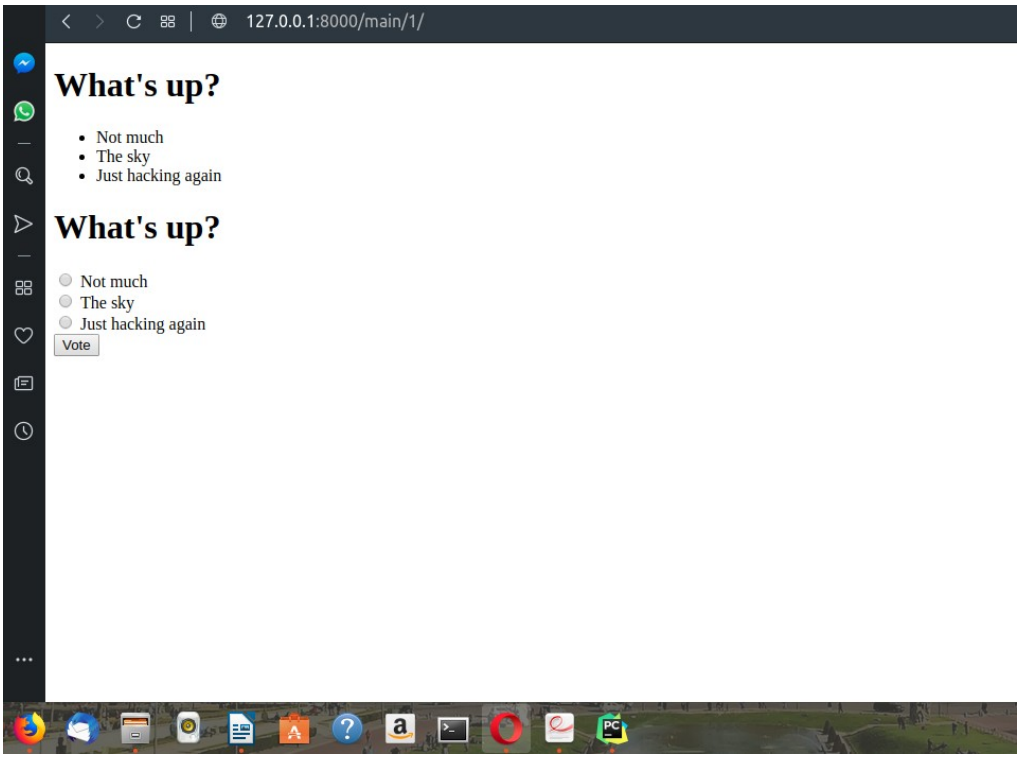
```
The first random generated array is:
49 44 26 32 17 45 5 14 30 32 23 30 16 45 8 15 14 15 40 9 16 14 24 8 47 25 40 23 24 17 11 23 13 39 5 32 36 13 47 16 45 20 49 11 15 9 28 30 27 18
The unique first random generated array is:
5 8 9 11 13 14 15 16 17 18 20 23 24 25 26 27 28 30 32 36 39 40 44 45 47 49
The second random generated array is:
39 43 34 16 3 31 43 46 4 17 13 17 40 28 9 46 10 45 9 7 14 6 28 15 19 45 24 47 27 1 17 17 47 1 35 2 33 28 48 39 47 13 7 37 41 16 35 4 13 46
The unique second random generated array is:
1 2 3 4 6 7 9 10 13 14 15 16 17 19 24 27 28 31 33 34 35 37 39 40 41 43 45 46 47 48
The minus result is:
-24 -60 -72 -105 -133 -147 -144 -156 -120 -128 -175 -273 -287 -264 -100 0 0 61 65 -140 -296 -231 -415 -425 -528 -552 2025 2116 2209 2304
```

سوال ۸:

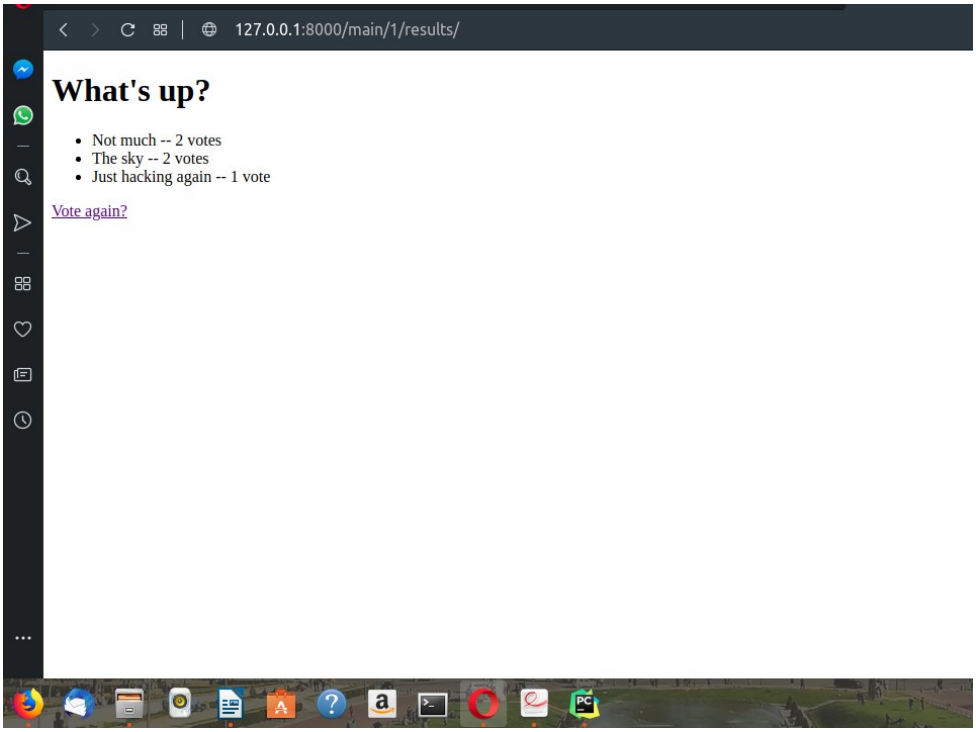
برای این سوال از documentation جنگو استفاده شد و مشابهت خروجی نیز به همین دلیل است. به طور مثال یک لینک در index.html قرار داده شده است تا کاربر با کلیک بر روی آن به صفحه رای گیری ارجاع داده شود.



صفحه رای گیری به شکل زیر است:



حال با کلیک بر روی هر کدام از گزینه ها و با زدن کلید vote رای ها در دیتابیس ذخیره می شوند و نتایج به صورت زیر در صفحه results به صورت زیر نمایش داده می شود:



با کلیک بر روی گزینه vote again هم دوباره کاربر به صفحه رای گیری بازگردانده می شود.