資料結構與演算法　　作業四

b06303077 經濟三 江雨柔

第一題

第一個 carArray 是一個一維動態陣列，指向的為此陣列的指標，長度為n的動態陣列，當宣告此陣列時會有n個car被創造出來，且只能呼叫到default constructor，無法將car初始化。

第二個 carArray 為二維動態陣列，指向的是指標的指標可以控制car的初始化。

第二題

直接將 void Entity::print() 改成與 void passenger::print() 相同的程式即可。因為現在 carArray 與 passengerArray 皆是繼承EntityArray,且print並無另外再定義，所以直接從Entity的print去做修改即可。

(原圖)

```
void Entity::print()
{
  cout << this->id << ": " << this->isOn << " " << this->isSer
       << " (" << this->lon << ", " << this->lat << ")" << endl;
}
```

(修改後)

```
Entity::Entity(string id, bool isOn, bool isSer, double lon, double lat)
  : id(id), isOn(isOn), isSer(isSer), lon(lon), lat(lat)
{
}

void Entity::print()
{
  cout << this->id << ": ";
  if(this->isOn == true)
  {
    if(this->isSer == true)
      cout << "in-service, (";
    else
      cout << "empty, (";
    cout << this->lon << ", " << this->lat << ")";
  }
  else
    cout << "offline";
  cout << endl;
}
```

第三題

EntityArray 只有繼承parent也就是entity的函式，所以他的print( )無法正確的將status印出來，而只是原封不動的將讀取的數字印出來，透過template來定義不同的typename 能讓 EntityArray 繼承到car 或是passenger的print，因此印出我們想要的結果。

第四題

若有許多帳號，有可能會超出20000這個限制，原本只會與其他錯誤一起顯示。但加上了throw，就能準確的判斷出是否超出陣列可儲存的空間。並同時告訴使用者與開法者這個程式已經達到了極限，而不是出現其他錯誤。

第五題

(a)

```
23
24  int main()
25  {
26    double grades[100] = {0}, avg = 0;
27    int gradeCnt = 0;
28    int total =0;
29    while(true)
30    {
31      double temp = 0;
32      cin >> temp;
33      if(temp == -1)
34        break;
35      else
36      {
37        total += temp;
38        grades[gradeCnt] = temp;
39        gradeCnt++;
40      }
41    }
42
43    if(total==0){
44      avg =0;
45    }else
46    {
47      avg = averageNonzero(grades, gradeCnt);
48    }
49
50    cout << avg;
51    return 0;
52  }
53
```

(b) 對於開發者來說第二項比較好，只要在main function裡面寫fail-safe的函式即可。如果今天有很多不同的函式要處理，每一條都列上 throw的話最後有可能忘記加上try-catch。所以統一在main 裡面判斷的話可以省去這種麻煩，也可以在 avg這個函式做出其他偵測錯誤的函式。

第六題

(a) 第七題中使用的是vector的 EntityArray，我們不在需要constructor來將陣列初始化，因此我們也不在需要destructor。由於使用的是vector不會有overflow的問題，所以在此可不用使用 exception handling。這樣做是合適的。

(b)

```
template <typename entityType>
int EntityArray<entityType>::getter()
{
  return entities.size();
}
```