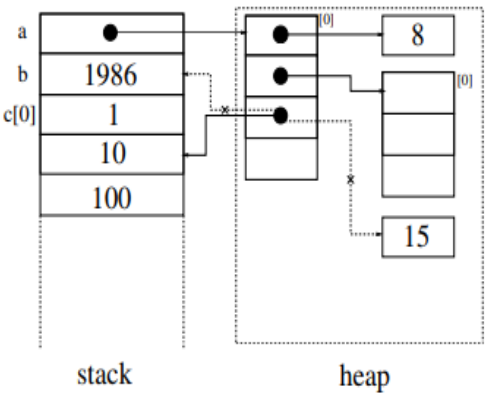
Lab section 13, TA: Stephane, Mentor: Wilson, Casey, Terry & Tyler.

```
int **a;
a = new int*[4];
a[0] = new int;
a[1] = new int[3]
a[2] = new int;
int b = 25;
int c[3] = {1,10,

*(a[2]) = 15;
a[0][0] = 8;
a[2] = &b;
a[2][0] = 1986;
a[2] = &(c[1]);
```

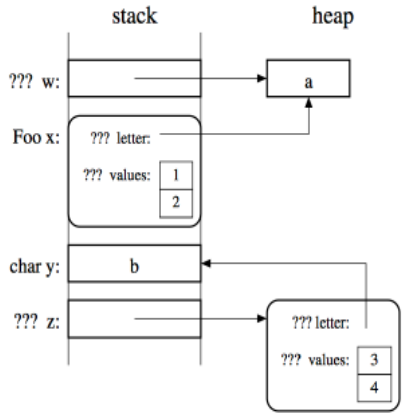


Solution:

```
class Foo {
public:
  Foo(char* l);
private:
  char* letter;
  int values[2];
};

Foo::Foo(char *l) {
  static int id = 1;
  letter = l;
  values[0] = id;
  values[1] = id+1;
  id += 2;
}

int main() {
  char* w = new char;
  *w = 'a';
  Foo x(w);
  char y = 'b';
  Foo* z = new Foo(&y);
}
```



-If using members in the class, u have to use

Dishawasher:: before the function.

Otherwise writ it outside private and public.

- When passing in vecters, strings, should use &

Since vectors and strings take more memories,

we don't want to make extra copies of them

int *p;

declares a pointer p points to NULL.

int *p = new int;

new creates a spot in heap

For ***classes***, DO NOT forget the ";" after declaring the class
The header file, file.h, contains the class declaration

The implementation file,
file.cpp, contains the member function definitions.
(Remember to #include "file.h")

To use a **default constructor**,
simply assign the variable without ()
`Ex. Name() ;`

Solution:

```
// CONSTRUCTOR
Dishwasher::Dishwasher(int max_
  cups = forks = knives = spoor
  max_plates = max_p;
}

// PRINT & ACCESSORS
void Dishwasher::printContents() const {
  std::cout << plates.size() << " plate(s):";
  for (int i = 0; i < plates.size(); i++) { std::cout << " " << plates[i]; }
  std::cout << std::endl;
  std::cout << cups << " cup(s)" << std::endl;
  std::cout << forks+knives+spoons << " utensil(s)" << std::endl;
}
int Dishwasher::completeUtensilSets() const {
  return std::min(std::min(forks,knives),spoons);
}

// MODIFIERS
bool Dishwasher::addPlate(const std::string& color) {
  if (valid_counts(plates.size()+1,cups)) {
    plates.push_back(color);
    return true;
  }
  return false;
}
bool Dishwasher::addCup() {
  if (valid_counts(plates.size(),cups+1)) {
    cups++;
    return true;
  }
  return false;
}
```

Solution:

```
class Dishwasher {
public:
  // CONSTRUCTOR
  Dishwasher(int max_plates);
  // PRINT & ACCESSORS
  void printContents() const;
  int completeUtensilSets() const;
  // MODIFIERS
  bool addPlate(const std::string& color);
  bool addCup();
  void addFork()  { forks++; }
  void addSpoon() { spoons++; }
  void addKnife() { knives++; }
private:
  // PRIVATE HELPER FUNCTION
  bool valid_counts(int p, int c) const;
  // REPRESENTATION
  int max_plates;
  std::vector<std::string> plates;
  int cups;
  int forks;
  int knives;
  int spoons;
};
```

- **const type& foo():** has to use with the type& => not going to change the return variable
- **foo() const:** cannot change the private variable of the class
(only used when as a member function in the class)
- **foo(const type& val):** Cannot change the val that is passed in.

Read / Write file from arguments:

- std::ifstream inFile(argv[#]) / std::ofstream outFile(argv[#])
- if (!inFile) {std::cerr << "ERROR!!" << std::endl;}

If a is an array. By simply writing a means the address of a[0].

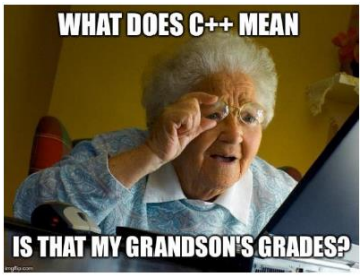
While deleting pointers:

Delete [] a represents deleting array a.

Delete [] a[1] represents deleting Whatever is a[1] points to.

Delete a only deletes first value in array a.

```
( )
6 10 8 + 9
12 16 x + y
{ }
[ ]
0 4 3 + 41
18 22 11 + 50
```



Solution:

```
//Could also use const std::vector<char>& for opens/closes. Need std::string for input
void PrintParses(const std::vector<std::string>& opens, const std::vector<std::string>& closes,
                 const std::vector<std::string>& input,
                 const std::vector<std::vector<unsigned int> >& parses){
    for(unsigned int i=0; i<opens.size(); i++){
        std::cout << opens[i] << " " << closes[i] << std::endl;
        for(unsigned int j=0; j<parses[i].size(); j+=2){
            std::cout << parses[i][j] << " " << parses[i][j+1];
            for(unsigned int k=parses[i][j]+1; k<parses[i][j+1]; k++){
                std::cout << " " << input[k];
            }
            std::cout << std::endl;
```

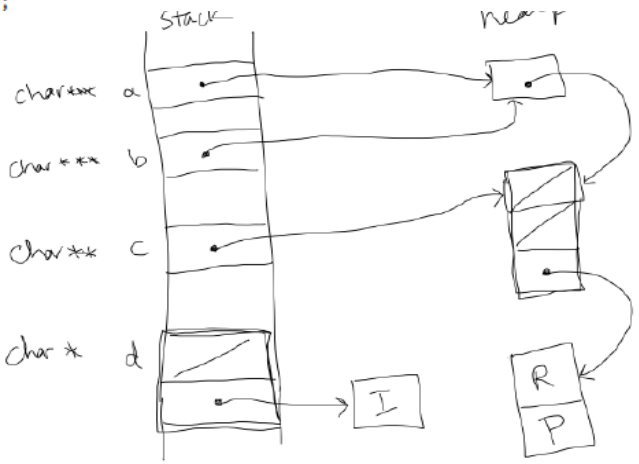
```
c[2][0] = 'R';
(*a)[0] = NULL;
(*b)[2][1] = 'P';
c[1] = NULL;
*(d[1]) = 'I';
d[0] = NULL;

// CREATE
// MEMORY
// DIAGRAM

delete [] (*a)[2];
delete b;
delete [] c;
delete d[1];
```

Solution

```
char*** a = new char**;
char*** b = a;
*b = new char*[3];
char** c = (*b);
c[2] = new char[2];
char* d[2];
d[1] = new char;
```



Solution:

```
class Customer {
public:
  // CONSTRUCTOR
  Customer(const std::string& name);
  // ACCESSORS
  const std::string& getName() const;
  const std::string& getStylist() const;
  const Date& lastAppointment() const;
  int numAppointments() const;
  // MODIFIERS
  void hairCut(const Date &d,const std::string &styli
private:
  // REPRESENTATION
  std::string customer_name;
  std::string preferred_stylist;
  std::vector<Date> appointments;
};

// helper function for sorting
bool stylist_then_last_appointment(const Customer &c1
```

.h file 里 private 用来声明 class 里可使用的 member variables

.cpp file 里的 Class::Class(){}初始化，把东西赋值给

Private 里的的变量供 class 来使用

Solution:

```
// open the file
std::ifstream istr("hw1_scores.txt");
assert (istr.good());
// variables to parse the file & store the data
std::string name;
int score;
std::vector<std::vector<std::string> > histogram;
// read the file
while (istr >> name >> score) {
  int bucket = score / 5;
  for (int i = histogram.size(); i <= bucket; ++i) {
    // add additional buckets if needed
    histogram.push_back(std::vector<std::string>());
  }
  // insert this student into the correct bucket
  histogram[bucket].push_back(name);
}
```

Next, write code to output the data stored in the vector to `std::cout` as shown above.

Solution:

```
// loop over all of the buckets
for (int i = 0; i < histogram.size(); i++) {
  // print the bucket range
  std::cout << "[" << std::setw(2) << i*5 << "-" << std::setw(2) << (i+1)*5-1 << "]";
  for (int j = 0; j < histogram[i].size(); ++j) {
    // print the names
    std::cout << " " << histogram[i][j];
  }
  std::cout << std::endl;
}
```