myStruct.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

const int SIZE = 100;

struct DateType {
  string month;
  int day;
  int year;
};

struct EventType {
  string place;
  DateType date;
};

void PrintStruct(DateType Holiday);
void AssignMonth(EventType Dates[]);
void PrintStructArray(EventType Dates[]);

int main() {
  DateType Holiday;
  Holiday.month = "January";
  Holiday.day = 5;
  Holiday.year = 2019;
  EventType Dates[SIZE];
  PrintStruct(Holiday);
  AssignMonth(Dates);
  PrintStructArray(Dates);
  return 0;
}

// pre  : Holiday is intialized with three field values
// post : Function prints date of Holiday in form mm dd, yy
void PrintStruct(DateType Holiday) {
  cout << Holiday.month << " " << Holiday.day << ", " << Holiday.year << "\n";
}


// pre  : none
// post : The month field of every date record in array is assigned a month
void AssignMonth(EventType Dates[]) {
  for(int i = 0; i < SIZE; i++) {
      Dates[i].date.month = "January";
  }
}


// pre  : Date filed of every date record is assigned a month.
// post : Function prints month of every date in the array.
void PrintStructArray(EventType Dates[]) {
  for(int i = 0; i < SIZE; i++) {
      cout << Dates[i].date.month << "\n";
  }
}
```

names.cpp

```cpp
#include <string>
#include <iostream>
using namespace std;

void BreakDown(string name, string& first, string& last, string& mi);

int main() {
  string name, first, last, mi;
  cout << "Name? <Last, First MI.> ";
  getline(cin, name);
  BreakDown (name, first, mi, last);
  cout << "First Name Entered :  " << first << endl;
  cout << "Last Name Entered :  " << last << endl;
  cout << "Middle Initial Entered :  " << mi << endl;
  return 0;
}

// pre  : name is initialized with a full name
// post : first, mi, and last contain the individual components of that name
void BreakDown (string name, string& first, string& mi, string& last) {
  int commaPos = name.find(",");
  int dotPos = name.find(".");
  last = name.substr(0, commaPos);
  mi = name.substr(dotPos - 1, 1);
  first = name.substr(last.length() + 2, name.length() - last.length() - 4);
}
```

Client program for Time class `testTime.cpp`

```cpp
#include <string>
#include <iostream>
#include "time.h"
using namespace std;

int main() {
  Time myTime(9,30,0);
  myTime.Write();
  myTime.WriteAmPm();
  myTime.Set(8,0,0);
  myTime.WriteAmPm();
  myTime.Mealtime();
  Time Schedules[10];
  for(int i = 0; i < 10; i++) {
    Schedules[i].Set(11,0,0);
    Schedules[i].WriteAmPm();
  }
  return 0;
}

void Time::WriteAmPm() const {
  bool am;
  int tempHrs;
  am = (hrs <= 11);
  if (hrs == 0) tempHrs = 12;
  else if (hrs >= 13) tempHrs = hrs - 12;
  else tempHrs = hrs;
  if (tempHrs < 10) cout << "0";
  cout << tempHrs << ":";
  if (mins < 10) cout << "0";
  cout << mins << ":";
  if (secs < 10) cout << "0";
  cout << secs;
  if(am) cout << " AM" << endl;
  else cout << " PM" << endl;
}
```

```
Time::~Time() {
  cout << "Destructor Called." << endl;
}

void Time::Mealtime() const {
  if(hrs == 8 && mins == 0 && secs == 0) {
    cout << "Breakfast" << endl;
  } else if (hrs == 12 && mins == 0 && secs == 0) {
    cout << "Lunch" << endl;
  } else if (hrs == 19 && mins == 0 && secs == 0) {
    cout << "Dinner" << endl;
  }
}
```

#mercer/spring2019/csc245