Quiz 2

cin.erase(); can delete a part of the string or clear all the contents of a string.

Quiz 3

Parent class - passes properties and functions to “derived” classes

Derived classes - can “inherit” from one or more baes classes.

Friend class - Designates that this “alien” class can access functions of another class

Host class - Class members must use class functions to access their property values.

Quiz 4

Stacks can only insert, remove and retrieve elements from the front of the structure

Some of stack’s stored elements may never be retrieved or accessed.

Queues are characterized as First In, First Out processing.

Stacks are not characterized as Last In, Last Out processing.

Newest stored elements are accessed first as Stack elements are accessed.

Stacks and Queues are “limited” in size.

```
struct node
{
   node *next = nullptr;
   int value;
   node *previous = nullptr;
};
//Used to Set up new First & Last Node
TEMP = new node;
TEMP -> previous = nullptr;
TEMP -> next = nullptr;
TEMP -> value = number;
FIRST = TEMP;
LAST = TEMP;
//Add First to List
TEMP = new node;
TEMP -> previous = nullptr;
TEMP -> next = FIRST;
TEMP -> value = number;
FIRST -> previous = TEMP;
FIRST = TEMP;
Size_nodes++;
//Add Last to List
TEMP = new node;
TEMP -> previous = LAST;
TEMP -> next = nullptr;
TEMP -> value = number;
LAST -> next = TEMP;
LAST = TEMP;
Size_nodes++;
//Traverse
TEMP = FIRST;
 for(int n = 1; n <= size_nodes; n++)
{
            Display(TEMP,n);
         TEMP = TEMP -> next;
}
TEMP = FIRST;
for(int n = 1; n <= size_nodes; n++)
{
            int Value = TEMP -> value;
         if(Value == number)
         {
                  Display(TEMP, n);
                  status = true;
                  Break;
            }
TEMP = TEMP -> next; }
}
```

```
//Read-in
ifstream InputFile;
   InputFile.open("States2.dat", ios::in);
   TEMP = new node;
   TEMP -> previous = nullptr;
   TEMP -> next = nullptr;
   FIRST = TEMP;
   LAST = TEMP;
   InputFile >> TEMP -> statename >> TEMP -> value;
   size_nodes++;
   while(!InputFile.eof())
   {
      TEMP = new node;
      TEMP -> previous = LAST;
      TEMP -> next = nullptr;
      InputFile >> TEMP -> statename >> TEMP -> value;
      LAST -> next = TEMP;
      LAST = TEMP;

      size_nodes++;
   }
   InputFile.close();

//Overwrite
ofstream outputFile;
outputFile.open("State2.dat", ios::out);

   TEMP = FIRST;
   for(int n = 1; n <= size_nodes; n++)
   {
      outputFile <<
      outputFile << "\n";
   }
   outputFile.close();

//Pure virtual (just a redirector)
//Must include in all inherited classes;
Virtual functions use pointers and Parent class datatype
Employee *worker = &Workers[0];

   virtual void terminate() = 0;
   virtual void calculatepay() = 0;
   virtual void Display(int n) = 0;
   virtual void setovertime() = 0;
//Use for inheritance.
Protected:
Employee *worker = &Workers[0];
```