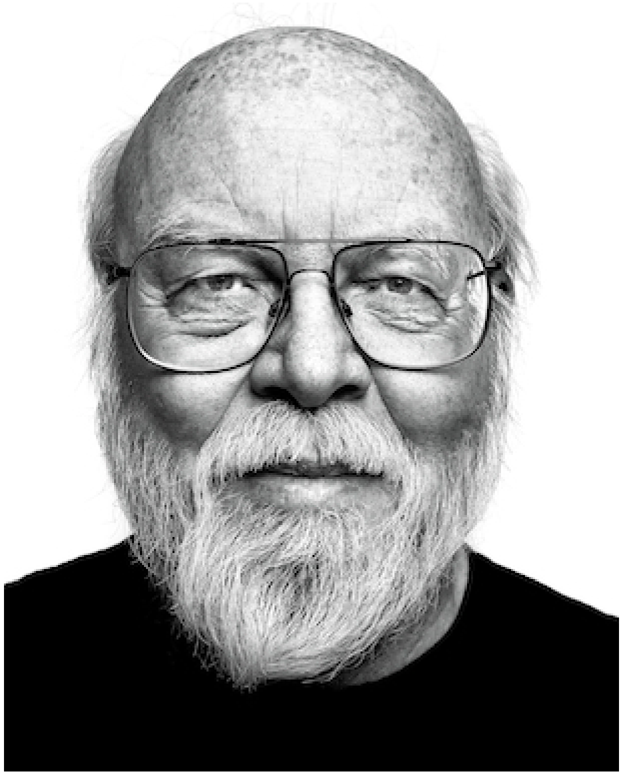
If you think there is a problem with a question or a question is unclear, email your instructor.

The use of the **auto** keyword is forbidden on this test.

Do not use any of the functions from `<algorithm>` unless the question asks for them.

Q2 Extra Credit
3 Points

Who is the creator of the Java programming language? Completely fill the circle next to your choice.



- Bill Gates
- Guido van Rossum
- Alan Turing
- Bjarne Stroustrup
- Claude Shannon
- Sergey Brin
- Ada Lovelace
- [✓ James Gosling]
- Dennis Ritchie
- None of the above

Q3

5 Points

What is the output of the following program?

```
class Television {
    virtual void connect() {
        cout << "scanning channels...";
    }
}

class Computer {
    virtual void connect() {
        cout << "joining network...";
    }
}

class SmartTelevision: public Television, public Computer {}

int main() {
    SmartTelevision x287;

    x287.connect();
}
```

- ○ The program compiles, runs, and outputs `scanning channels...`
- ○ The program compiles, runs, and outputs `joining network...`
- ○ The program compiles, runs, and outputs `scanning channels...joining network...`
- ○ The program compiles, runs, and outputs `joining network...scanning channels...`
- [✓ The program fails to compile.]
- ○ The program compiles and crashes when it runs.
- ○ None of the above

Q4
5 Points

Given an STL `list` containing string objects named `all_strings`, find the longest string in the list and output this string. **You can assume that no two strings in the list will be the same length.**

You do not have to write a function or even define the list, simply use `all_strings`.

- Do not use auto
- Do not use a ranged for loop

REMEMBER TO USE SPACES TO INDENT YOUR CODE!

```
list<string> all_strings{};
   string longest_str = "";
   for(list<string>::iterator iter = all_strings.begin(); iter !=
all_strings.end(); iter++){
      if((*iter).size() > longest_str.size()){
         longest_str = *iter;
      }
   }
   cout << "Longest word: " << longest_str << endl;
```

Q5

5 Points

Disappointed with the lack of a dedicated exponentiation operator, a C++ developer decides to implement one for numeric types using the symbol `@=`. To start, a custom integer class named `Integer` has been defined in a header file named `Integer.h` as follows:

```
class Integer {
public:
  Integer(int val) {}                          // Line 1

  Integer& operator@=(int power) {             // Line 2
    int orig = val;                            // Line 3
    for (int i = 0; i < power; ++i) {          // Line 4
      val *= orig;                             // Line 5
    }
    return *this;                              // Line 6
  }

private:
  int val;                                     // Line7
};
```

When properly including this header file in a program using the directive `#include "Integer.h"` and compiling the program, what would happen?

- ○ Compilation error at Line 1
- [✓ Compilation error at Line 2]
- ○ Compilation error at Line 3
- ○ Compilation error at Line 4
- ○ Compilation error at Line 5
- ○ Compilation error at Line 6
- ○ Compilation error at Line 7
- ○ No compilation errors from `Integer.h`
- ○ None of the above

Q6

5 Points

What would be output by the program below?

```
void increment(int* num, int n_times) {
  for (int i = 0; i < n_times; i++) num = num + 1;
}

int main() {
  int val = 2;

  increment(&val, 3);

  cout << val;
}
```

- 4
- 5
- 6
- [✓ 2]
- 8
- The program fails to compile
- The program compiles and crashes when it runs
- None of the above

Q7

5 Points

What would be output by the program below?

```
class Square {
public:
  Square(int side_length = 1) : length(side_length) {}

  virtual double calc_area() const {
      return length * length;
  }
protected:
  double get_length() const { return length; }

private:
  double length;
};

class Rectangle: public Square {
public:
  Rectangle(double width = 1) : width(width) {}

  double calc_area() const {
      return get_length() * width;
  }

private:
  double width;
};

void display_area(Square sq) {
  cout << sq.calc_area();
}

int main() {
  Rectangle rect(4);

  display_area(rect);
}
```

- ○ Undefined
- ○ 0
- [✓ 1]
- ○ 4
- ○ 5
- ○ The program fails to compile
- ○ The program compiles and crashes when it runs
- ○ None of the above

Q8
8 Points

Provide a definition of a struct `LessThan` and templated function `replace_if` as described in parts A and B, respectively, to enable the code below to work as expected. Consider the code below:

```
LessThan pred_func(3);

vector<int> my_ints { 1, 7, 0, 2, 8, 5, 3, 6, 2, 4 };
replace_if(my_ints, my_ints + 10, pred_func, -1);
```

After the above code is executed, all values less than or equal to `3` are replaced with `-1` in `my_ints`, such that `my_ints` contains the integers: `-1 7 -1 -1 8 5 3 6 -1 4`

Q8.1 Part A

4 Points

Define the type `LessThan` which overloads the call operator, `()`, and has a single integer member variable. When instantiated this type's call operator accepts an integer parameter and returns a boolean indicating whether the integer is less than the member variable's value.

REMEMBER TO USE SPACES TO INDENT YOUR CODE!

```
struct LessThan{
        LessThan(int val): value(val){}
        bool operator()(int num) const{
               return num < integer;
  }
  int val;
};
```

Q8.2 Part B

4 Points

Write a function template (also known as a templated function) that will act like the STL's generic algorithm `replace_if`, which assigns a value passed as the function's 4th argument to all the elements in the half-open range that satisfy the predicate (a function object) passed as the

3rd argument. Note that this function must be implemented such that the half-open range can be defined using various types (list, vector, array, etc) and the range can contain values of any type.

REMEMBER TO USE SPACES TO INDENT YOUR CODE!

```
template<typename T, typename U, typename V>
void replace_if(T start, T end, U pred, V target) {
          for (T iter = start; iter != end; ++iter) {
                 if (pred(*iter) == true) {
                     *iter = target;
                }
          }
  }
```

Q9
5 Points

Given the following program,

```
class GreetingCard {
public:
  GreetingCard(double price, string msg = ":-)") :
      price(price), msg(msg) { }        // Line 1

  void display() {
    cout << '$' << price << " - " << msg; // Line 2
  }

private:
  double price;
  string msg;
};

int main() {
  GreetingCard card(3.99, "Happy Birthday!"); // Line 3
  card = 5.99;           // Line 4
  card.display();        // Line 5
}
```

What will be the result of compiling and running the program?

- ○ Compilation error at Line 1
- ○ Compilation error at Line 2
- ○ Compilation error at Line 3
- ○ Compilation error at Line 4
- ○ Compilation error at Line 5
- [✓ The program compiles, runs, and outputs `$5.99 - :-)`]
- ○ The program compiles, runs, and outputs `$5.99 - Happy Birthday`
- ○ The program compiles, runs, and outputs `$3.99 - Happy Birthday`
- ○ The program compiles, runs, and outputs `$3.99 - :-)`
- ○ The program compiles and crashes when it runs
- ○ The program fails to compile
- ○ None of the above

Q10

5 Points

What is the result of compiling and running the program below?

```
class Pet {
public:
  virtual void communicate() const = 0; // Line 1
};

class Cat : public Pet {
public:
  void communicate() const {
    cout << "meow" << endl; // Line 2
  }
};

class Dog : public Pet {
public:
  void communicate() {
    cout << "woof" << endl; // Line 3
  }
};

void pet(const Pet& p) {
  p.communicate();      // Line 4
}

int main() {
  Dog fido;      // Line 5

  pet(fido);     // Line 6
}
```

- ○ Compilation error at Line 1
- ○ Compilation error at Line 2
- ○ Compilation error at Line 3
- ○ Compilation error at Line 4
- [✓ Compilation error at Line 5]
- ○ Compilation error at Line 6
- ○ The program compiles, runs, and outputs woof
- ○ The program compiles, runs, and outputs meow
- ○ The program fails to compile
- ○ The program compiles and crashes when it runs
- ○ The program compiles, runs, and does not output anything
- ○ None of the above

Q11

5 Points

What is output by the program?

```
class Ancestor {
public:
  virtual void reveal() { cout << "Ancestor"; }
};

class Parent : public Ancestor {
public:
  void reveal() { cout << "Parent"; }
};

class Child : public Parent {
public:
  virtual void reveal() { cout << "Child"; }
};

void meet(Ancestor& a, Parent& p) {
  a.reveal();
  cout << " <-> ";
  p.reveal();
}

int main() {
    Child ch;
    Ancestor an;

    meet(an, ch);
}
```

- ○ Parent <-> Parent
- ○ Parent <-> Child
- [✓ Ancestor <-> Child]
- ○ Ancestor <-> Parent
- ○ Ancestor <-> Ancestor
- ○ Child <-> Child
- ○ The program fails to compile
- ○ The program compiles and crashes when it runs
- ○ None of these

Q12

12 Points

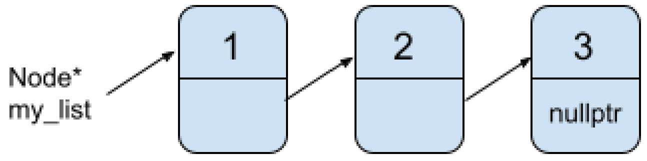
Write a function `split_list` that is passed a linked list and a target value to remove. The function will remove (and free memory) for all the Nodes in the list that contain the target value. Any `Node` preceding the removed `Node` will become the tail of a linked list. Any `Node` that follows the removed `Node` and does not contain the target value, will be the head `Node` of a linked list. A pointer to this head `Node` will be added to a vector containing all of the resulting linked lists that are split from the original linked list. Other Nodes will not be modified. You can assume that the target value will never appear in consecutive nodes in the list.

By linked list we mean a `Node` pointer, where the type `Node` is defined with:

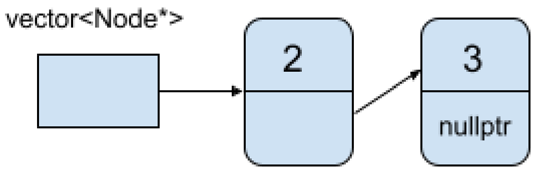
```
struct Node {
  Node(int data = 0, Node* next = nullptr)
    : data(data), next(next) {}
  int data;
  Node* next;
};
```

Example 1

Assume the function `split_list` is passed the following linked list and the target value of 1 (e.g. `split_list(my_list, 1)`):

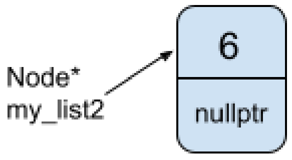


After calling the function, `split_list` would return a new vector (size of 1) as represented below:



Example 2

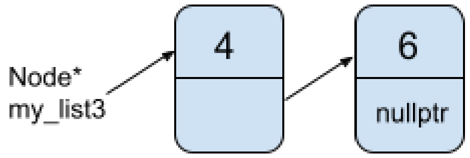
Assume `split_list` is passed the following linked list and target value of 6 (e.g. `split_list(my_list2, 6)`):



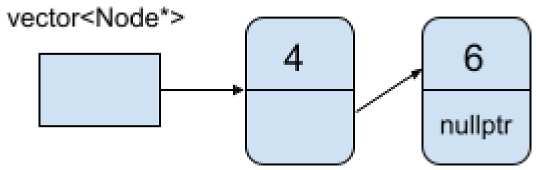
After calling the function, `split_list` would return a new vector that is empty (size of 0).

Example 3

Assume `split_list` is passed the following linked list and target value of 0 (e.g. `split_list(my_list3, 0)`):

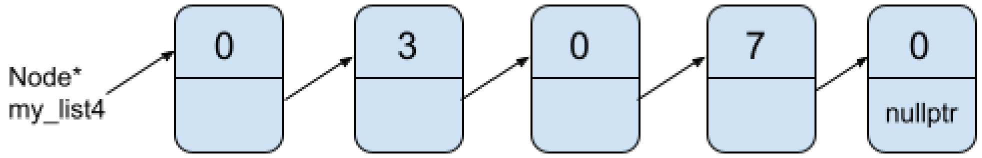


After calling the function, `split_list` would return a new vector (size of 1) as represented below:

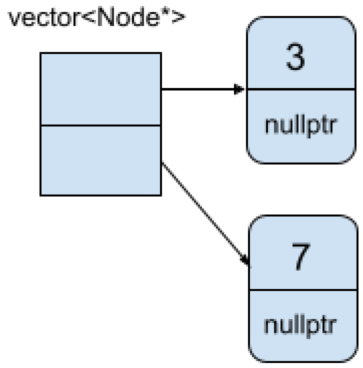


Example 4

Assume `split_list` is passed the following linked list and target value of 0 (e.g. `split_list(my_list4, 0)`):



After calling the function, `split_list` would return a new vector (size of 2) as represented below:



Example 5

In the case that `split_list` is passed an empty list (e.g. `split_list(nullptr, 8))`, the function will return a new vector that is empty (size of 0).

REMEMBER TO USE SPACES TO INDENT YOUR CODE!

```
vector<Node*> split_list(Node* head, int target){
       vector<Node*> lists;
        Node* ptr = head;
         while(ptr!=nullptr){

            if(ptr->data == target){

                  if(ptr->next == nullptr){
                        return lists;
                  }

                  Node* new_head = ptr->next;
                  lists.push_back(new_head);
                  delete ptr;
                  ptr=new_head;
            }
       else
               {
                  ptr=ptr->next;
            }
      }
      return lists;
}
```

Q13

10 Points

Based on our `Vector` class which only held integers, we want to implement a class `Iterator`, sufficient to handle a ranged-for. (You are not writing any part of the class `Vector`.)

Our goal is for the following function to work properly, i.e. to add `1831` every `int` in the `Vector` `vec`.

```
void increaseValues(Vector& vec) {
    for (int& val : vec) {
        val += 1831;
    }
}
```

As you know, the `Vector` class must support the methods `begin()` and `end()`, returning appropriate instances of your `Iterator` class. The `begin()` method will initialize its `Iterator` with a pointer to the beginning of the dynamic array and the `end()` method will initialize its `Iterator` with a pointer just past the last item. But this is not your job!

It is up to you to define your `Iterator` class appropriately with any functions needed. Note that you do not have to know anything more about the `Vector` class, nor do you need access to any of its methods or fields.

REMEMBER TO USE SPACES TO INDENT YOUR CODE!

```
class Iterator{

    friend bool operator==(const Iterator& lhs, const Iterator& rhs){
        return (lhs.ptr == rhs.ptr);
    }
public:
//constructor
    Iterator(int* ptr): ptr(ptr){}

    //methods

    //pre inc
    Iterator& operator++(){
        ++ptr;
        return *this;
    }

    //post inc
    Iterator operator++(int dummy){
        //copy
        Iterator original(*this);
        //overload
        ++(*this);
        return original;
    }

    //deref
    int& operator*() {return *ptr;}

private:
    int* ptr;
};
bool operator!=(const Iterator& lhs, const Iterator& rhs){
        return !(lhs==rhs);
}
```

Q14

30 Points

In this question you will be modeling a university setting with its community members (faculty and students). Yes, the question involves inheritance.

A `University` consists of a collection of community `Member`s. There are various types of `Member`s, but you will only be defining the `Faculty` and `Student` types.

We will model the `University`'s collection of faculty and students by writing an `add_member` method in the `University` class to add `Member`s to the university. `Member`s that are being added to the collection are not permitted to already be in the community of another university. The `add_member` method should not fail silently!

Prior to its opening each day, the `University` has a community security check by calling the `University` class' `security_check` method. The method will in turn call upon each `Member` to print its security level using the `Member`'s `check_security` method. Note, every type of `Member` must implement their own `check_security` method.

- All `Member`s have names.
- `Student`'s security values are provided when they are created. (See the test code.)

- `Faculty` do not have security values, by default they have a high trust level.

Your task is to define the following classes: `University`, `Member`, `Faculty`, and `Student`.

Hint: this question has nothing to do with templates or copy control.

Example Test Code

Below is an example of using your classes. Be sure that your class definitions will allow this test code to work.

```
int main() {
    University nyu;
    University columbia;

    Faculty depasquale("Peter DePasquale");
    Faculty sterling("John Sterling");
    Faculty reeves("Darryl Reeves");
    Student chen("Mike Chen", 100);
    Student smith("Sarah Smith", 100);
    Student liu("Christie Liu", 200);

    nyu.add_member(depasquale);
    nyu.add_member(sterling);
    nyu.add_member(reeves);
    columbia.add_member(reeves);

    nyu.add_member(chen);
    nyu.add_member(smith);
    nyu.add_member(liu);

    nyu.security_check();
}
```

Example Output

```
Peter DePasquale: has high level security.
John Sterling: has high level security.
Darryl Reeves: has high level security.
Mike Chen: has security level: 100
Sarah Smith: has security level: 100
Christie Liu: has security level: 200
```

REMEMBER TO USE SPACES TO INDENT YOUR CODE!

```
class University;


class Member{
    public:
    //constructor

    Member(const string& name): name(name){
        community = nullptr;
    }

    //methods
    virtual void check_security() const = 0;
    void set_university(University& university) {
        community = &university;
    }
   const string& getName() const { return name; }

   const University* get_university() const { return community; }

    private:
    string name;
    University* community;

};// class end


class University {
    public:
    //methods
    bool add_member(Member& member) {
        if (!member.get_university()) {
            vec.push_back(&member);
            member.set_university(*this);
            return true;
        }
        return false;
    }

    void security_check() const {
        for (const Member* member: vec) {
            member->check_security();
        }
    }



private:
    vector<Member*> vec;
};




class Faculty : public Member {
public:
    Faculty(const string& name) : Member(name) {}

    void check_security() const {
        cout << getName() << ": has high level security." << endl;
    }
};// class end

class Student : public Member {
public:
    Student(const string& name, int security_num) : Member(name),
security_num(security_num) {}

    void check_security() const {
        cout << getName() << ": has security level: " << security_num
<< endl;
    }

private:
    int security_num;

}; // class end
```