1. Given the following code fragment:

```
for (j=0; j<3; j++)
          for (k= 0; k<=j; k++)
              for (i=1; i< k; i++)
                  cout << "Good Luck" << endl;
```

(a) How many times does "Good Luck" get printed.

```
It prints once for each i value.
If the inner most loop conditions are:
i=1; i<k  then the number of prints is  1
i=0; i<k                                4
i=0; i<=k                               10
```

2. Given the following code fragment.

```
int main()
    {        int p;
             cin >> p ;
             k = power(2,p);  // this computes 2^p
             f(2, k );
    }

void f(int j, const int r)
  {
    if (j >= r) return;
       else { cout <<"ship ahoy!" << endl;
                f (2*j,r);
              }
 }
```

(a) How many times does "ship ahoy" get printed?

```
It prints once for each value j < r = 2^p.
j = 2, 2^2, 2^3, ... , 2^(p-1)
So, p-1 times.
```

3. Cicle which function is larger, *f* or *g* (i.e., which function grows larger for all values of n).

(a) **f(n)** = n g(n) = nlogn `g(n) grows faster.`

4. Show a diagram of memory after the execution of each statement and *give the output of the code fragment*. Assume the following declarations:

```
float i, x, *p, *q;

p = &i;

*p = 10.0;

 q = &x;

*q = 8.6;

  cout << x << endl;
  cout << I << endl;
```

```
8.6
10.0
```

```
*p = 9.5;

p = q;

q = &i;

cout << *p <<endl;

cout << i <<endl;

cout << x << endl;

cout << *q << endl;
```

```
8.6
9.5
8.6
9.5
```

6. What gets printed from the following program?

```
#include <iostream>

using namespace std;
void f (int);

int main ()
     {
      f(3);
      return 0;
      }



void  f(int n)
   {
       static int x = 1;
       if (n <=1) return 0;

         f (n-1);
         x++;
         f (n-2);
         x++;

         cout << "x = " << x << "   n =  " << n << endl;

         return 0;
   }
```

```
Output:
x=3 n=2
x=5 n=3
```

7. Given an array of names (stored as pointers to strings), write a c++ program to print out the "mode", i.e., the name that appears the most times in the array and how many times it appears. Please make sure to explain each step what you are doing in comments before the code. ***You must write out all of the code, no other functions are defined for you.*** **You can make one of the following assumptions:**

- ***(a) assume there is only one "mode"***
- ***(b) (***extra credit) assume there can be more than one "mode"***

```
int main ()
 {

char *names[100];

//assume this initializes the array with 100 names in arbitrary order, the names can be any length.
//You do not have to write the code for this function.
initialize (names);

//put your code here, please make sure you explain what you are doing in comments
```

8. Given a stack s which contains the following elements: (2, 4, 6, 8, 10, 12 14) where 2 is at the bottom of the stack and 14 is at the top,
(a) what does the routine, ***broccoli(),*** **print when called in the main program as shown below.**
(b) What does the stack s look like after execution of the routine broccoli?

```
  main ()
    {
       broccoli(s);

  }

void  broccoli(stack &s)
    {
        int i;
        if (s.isempty) return;
        i = s.pop();
        broccoli (s);
        cout << i << endl;
        s.push(i);
        return;
    }
```

```
Output:
2
4
6
8
10
12
14
```

```
The stack is restored
as the original.
```

9. Given the following program fragment:

```
int i, j, x;
cin >>x;
i = 1; j =1;
while (i < 10) {
   j = j*i;
   i++;
   If (i == x) break;
  }
```

Which of the following statements about the variables ***i*** and ***j*** must be true after the execution of the program fragment:

(a) j = (x-1)! **and** i >= x
(b) j = 9! **and** i = 10
(c) j = 10! **and** i = 10
(d) (j = 10! **and** i = 10) **or** (j = (x-1)! **and** i = x)
(e) (j = 9! **and** i >= 10) **or** (j = (x-1)! **and** i = x)

Answer: e

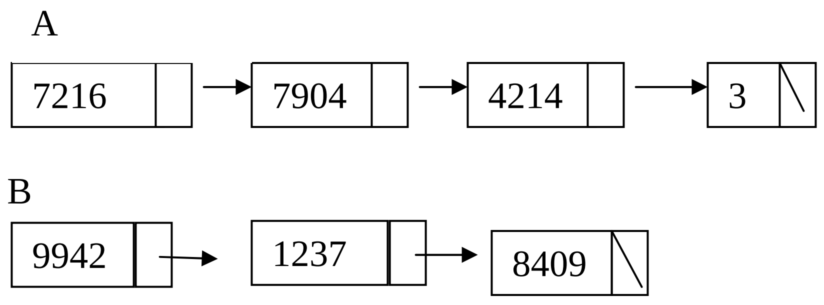
12. One way to store a very-long integer is to treat it as a *base-10,000* number, i.e., each base$_{10,000}$ digit would be a value in the range of 0 to 9999. A long integer can easily be broken up by grouping every 4 decimal numbers. For example, the number 3,421,479,047,216 would be represented by four base$_{10,000}$ digits:

digit 1	*digit 2*	*digit 3*	*digit 4*	
3	4214	7904	7216	**A**

and the number 840,912,379,942 would look like:

digit 1	*digit 2*	*digit 3*	
8409	1237	9942	**B**

We can use a linked list for this representation. Consider the addition of two polynomials, A+B. Addition proceeds from the least significant digit to the most significant. In terms of the linked list, this would mean traversing from the end of the linked list to the front. Since it is inefficient to keep traversing the linked list backwards (as our linked list only has a head pointer), the algorithm would be simpler if numbers were stored with the least significant component first, i.e., the base$_{10,000}$ digits should be stored in reverse order. So, A and B above would look like:



Using the definition of the List below, please fill in the functions specified by the function headers on the next page.

```
class Long_Integer //from Carrano – without exception handling
{
public:
    Long_Integert();
   Long_Integer(const Long_Integer& aList);
   ~Long_Integer();
// list operations:
   bool isEmpty() const;
   int getLength() const;
   void insert(int index, int newItem);
   void printlist();
   void retrieve(int index, ListItemType& dataItem) const;
   operator + (Long_Integer & x);
private:
   struct ListNode
   {
      int item;
      ListNode *next; };
   int size;
   ListNode *head;
   ListNode *find(int index) const;
   // Returns a pointer to the index-th node in the linked list.
```

1. Show the output produced by the following C++ program:

```
#include <iostream>

class BaseClass
{   public:
        int f() {return 1;};
        virtual int g() {return 2;};
};

class DerivedClass : public BaseClass
{   public:
        int f() {return 3;};
        int g() {return 4;};
};

int main()
{    BaseClass *b;
     DerivedClass *d;
     b = new DerivedClass;
     d = new DerivedClass;
     std::cout << b->f() << b->g() << d->f() << d->g();

     delete b;
     delete d;

     return 0;
}
```

```
Output:
1434
```

2. Suppose we have the following pair of class definitions, along with implementations of some functions of the derived class.

```
class Odd {
 public:
 virtual void setName(string name);
 virtual void setID(int id) = 0;

 protected:
 int id;
 private:
 string name;
};

class Deranged: public Odd {
 public:
 virtual void setName(string name);
 virtual void setID(int id);
};

 Deranged::setName(string name) {
 this->name = name;   // this is statement A
 }

Deranged::setID(int id) {
 this->id = id;              // this is statement B
 }
```

The question concerns whether the two statements labeled A and B are legal (i.e., will be accepted by a standard C++ compiler).

(a) Both statements are legal
(b) A is legal, B is not
(c) B is legal, A is not
(d) Neither statement is legal

Answer: (c)

3. Show a figure after the execution of each statement. Assume the following type definition:

```
class Anode {
public:
   int info;
   Anode *next;
};

Anode *p, *q, *r, *s, *t;

 p = new Anode;

    r= p;
    p->info = 5;
    r->info = 6;

    cout <<r->info;                    66
    cout <<p->info;

    r = new Anode;
    r->info = 7;
    p->next = r;
    s=p;
    q = new Anode;
    q->info = 8;
    r->next = q;
    p=p->next;

    cout << p->info <<endl;          7
    cout << q->info <<endl;          8
    cout << r->info <<endl;          7
    cout << s->info <<endl;          6


    s->next = r->next;
    s = s->next;

    cout <<s->info <<endl;          8
    cout <<r->info <<endl;          7
```

4. Given the following class:

```
  class point {
   public:

     float x, y;

     point() { x=0; y=0; }
        Ans (1): point(int ix, int iy): x(ix), y(iy) {}
     point operator * (point & pt);

  };
```

(1) (2 points) Modify the above definition of class point so that one can declare objects of class point with coordinates (x, y). Please do the modifications in-line.

(2) (5 points) Overload the operator * such that if a point object p1 with coordinates (x1,y1) and a point object p2 with coordinates (x2,y2) are multiplied, then the product p1*p2 has coordinates (x1*x2 + y1*y2 , x1*x2 - y1*y2).

```
Ans (2):
point point::operator * (point & pt)
   {
     point temp;
     temp.x = x*pt.x + y*pt.y;
     temp.y = x*pt.x - y*pt.y;
     return temp;
   };
```

5. Consider the following C++ function:

```
#include <iostream>
#include <vector>

using namespace std;

void Foo(int i)
{
    vector<int> *ptrToVector = new vector<int>(i);
    if (i % 2 == 1)
      return;  replace "return" with {delete pteToVector; return;}

    // do something useful

    delete ptrToVector;
}
```

Does function Foo cause a memory leak? If so, correct the problem. (You only need to show the part(s) being corrected.)