MAKE SURE YOUR NAME IS ON THE TOP OF EACH SHEET.

- All functions must be properly commented.
- You do not need to do any error checking within your functions.
- The back of the exam contains several pages of useful information.
- You may just write the body of functions whose headers are given (there is no reason to rewrite the header and the comments).
- There is an extra credit problem which can be used to replace one of the other programming problems, or to make up for points missed elsewhere. You may not get more than 100 points total. Save this problem for last, it is harder and partial credit will be harder to obtain.
- If you have no idea where to start on a problem, you may request a hint. Getting a hint means you can receive approximately half credit on that question (which is better than nothing).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Value</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Extra</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 -- 10 points -- Definitions

Define 5 of the 6 terms below. You do not need a formal definition, just a good description. Limit your answers to less than 50 words per items. You may use an example if it would be helpful. Clearly mark the term you do not wish to include; otherwise I will discard the final term.

a. Flag:

b. Call By Value:

c. Default Constructor:

d. Loop Invariant:

e. Overloaded Functions/Operators:

f. List the 3 properties of a Vector:
Problem 2 -- 10 points -- Matrices

Given the matrix, `mat` diagramed below:

```
mat:
1  3  5  7
0  1  2  3
8  6  4  2
4  5  6  7
```

what is the state of `mat` after the following code fragment has been executed. Fill in the blank matrix below. Show work to receive partial credit:

```c
for (k=0; k<4; k++)
{
    for (j=k; j>=0; j--)
    {
        mat[j][k] = mat[k][j]+k;
    }
}
```

```
mat:
```

```c
for (k=0; k<4; k++)
{
    for (j=k; j>=0; j--)
    {
        mat[j][k] = mat[k][j]+k;
    }
}
```
Problem 3 -- 10 points -- Searching

Given the vector \( v \) diagramed below:

\[
\begin{array}{cccccccccc}
1 & 3 & 8 & 11 & 12 & 23 & 31 & 34 & 39 \\
\end{array}
\]

what is the output of the following program fragments:

```cpp
int ans = PrintingBinSearch(v, 9, 34);
cout << "Solution:" << ans << endl;

ans = PrintingBinSearch(v, 9, 5);
cout << "Solution 2:" << ans << endl;
```

The function \texttt{PrintingBinSearch} is given below:

```cpp
int PrintingBinSearch(const Vector<int> & v, int numElts, int key) {
    int low = 0;
    int high = numElts - 1;
    int mid;
    while (low <= high) {
        mid = (low + high)/2;
        cout << low << " " << mid << " " << high << endl;
        if (v[mid] == key) {
            return mid;
        } else if (v[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}
```
Problem 4 -- 10 points -- Why??

a) The following function `ChopHalf` shown here is supposed to remove the first half of a list (you can assume the list always contains an even number of elements). For example, the list `(taylor buffett garcia clapton)` should be changed to `(garcia clapton)`. However, the list is changed to `(buffett garcia clapton)`. Explain why.

```c
void ChopHalf (List<string> & list)
{
    int k;
    for(k=0; k < list.Length()/2; k++)
    {
        list.ChopFront();
    }
}
```

b) Binary Search on a sorted vector is a very efficient way to search (logarithmic). Is Binary Search equally efficient in searching a sorted list of items? Why or why not?

**Hint:** Think about how are lists and vectors are different.
Problem 5 -- 10 points -- Arrange

Write the function `Arrange` which takes two integers and returns them in sorted order. For example:

```c
int x = 8;
int y = 6;
Arrange(x, y);
```

After this code has executed, `x == 6` and `y == 8`.

Note: `Arrange` must be a `void` function.
Problem 6 -- 15 points -- Snacks

Write the function Snacks whose prototype is given below. Snacks reads in product information from a stream in the form shown below:

- good: twinkies
- too greasy: pork rinds
- great: oreos and milk
- not bad: grapes
- too greasy: buttered popcorn
- great: phish food

If the data above is in the stream input, the call Snacks(input,"great") yields the following (you must number each line of output and give the total at the end):

1. oreos and milk
2. phish food
A total of 2 items is/are great

Complete the function Snacks:

```c
void Snacks (istream & input, string describe)
// purpose: To print all items which fit the description
// preconditions: The input stream is open and contains at least one line of input
// postcondition: The information is output to the screen
```
Problem 7 -- 15 points -- IsSorted

Write the function `IsSorted` whose prototype is shown below. `IsSorted` returns whether the list is sorted, regardless of capitalization. For example:

A: \((a \ A \ b \ F \ f \ G \ H \ j \ M \ m \ P \ P \ P)\)
B: \((A \ a \ B \ f \ E \ G \ H \ J \ M \ O \ P \ p)\)

`IsSorted(A)` returns `true`, but `IsSorted(b)` returns `false` since `f E` is not in sorted order.

```cpp
bool IsSorted(const List<char> & list)
// purpose: return true only if the list is sorted
//          (regardless of capitalization),
//          otherwise return false
// precondition: none
// postcondition: the correct value is returned
```
Problem 8 -- 20 points -- Library

The struct `CheckedOut` maintains the number of books a person has checked out from the library. The declaration is:

```cpp
struct CheckedOut
{
    string name;          // person’s name
    int number;           // number of books checked out
}
```

a) (10 points) Write the function `NumBooks` whose prototype is shown below. `NumBooks` returns the number of books a given person has checked out. Assume all names are lower case. Given the vector `v` diagramed below, the call `NumBooks(v,"jill",9)` returns 8 and the call `NumBooks(v,"jack",9)` returns 0.

```
int NumBooks(Vector<CheckedOut> & v, string who, int numElts)
// purpose: count the number of books who has checked out
// precondition: numElts <= v.Length()
// postcondition: the correct value is returned
```

<table>
<thead>
<tr>
<th>bill</th>
<th>will</th>
<th>jill</th>
<th>bill</th>
<th>lil</th>
<th>jill</th>
<th>will</th>
<th>bill</th>
<th>gil</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Problem 8 -- 20 points -- Library (Continued)

b) (10 points) Write the function MaxBooks whose prototype is shown below. MaxBooks returns the person who has the most number of books checked out. Assume all names are lower case. You must use the function NumBooks in MaxBooks -- you may assume that NumBooks works as specified even if yours does not. Given the vector v diagramed below, the call MaxBooks(v, 9) returns "bill".

<table>
<thead>
<tr>
<th>bill</th>
<th>will</th>
<th>jill</th>
<th>bill</th>
<th>lil</th>
<th>jill</th>
<th>will</th>
<th>bill</th>
<th>gil</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

string MaxBooks(Vector<CheckedOut> & v, int numElts)
// purpose: find who has the most books checked out
// precondition: numElts <= v.Length()
// postcondition: the correct value is returned
Extra Credit -- 10 points -- RemoveOdd

This problem is extra credit and can be done in place of another problem or to make up missed points (you can not get more than 100 points total).

Write the function `RemoveOdd` whose prototype is shown below. The function must remove all Odd numbers from the vector and update the number of elements remaining in the vector (it does not need to resize the vector).

Hints: Maintain a variable which maintains the current number of even items remaining in the vector. You should only use one loop (i.e. your algorithm must be linear).

Write the function `RemoveOdd` and list two valid postconditions.

```c++
void RemoveOdd(Vector<int> & v, int & numElts)
// purpose: As stated above
// preconditions: numElts <= v.Length()
// postconditions: All odd elements have been removed. All
//                 values between the final numElts value and the
//                 original numElts value are undefined.
```

```
<table>
<thead>
<tr>
<th>vect: 1 4 25 6 7 10 12 11 16 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>num: 9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>vect: 4 6 10 12 16 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>num: 5</td>
</tr>
</tbody>
</table>
```

<table>
<thead>
<tr>
<th>vect: 4 6 25 6 7 10 12 11 16 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>num: 9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>vect: 4 6 10 12 16 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>num: 5</td>
</tr>
</tbody>
</table>
The Vector Class:

// Vector() -- default, vector of size 0 (no entries)
// Vector(int size) -- vector with size entries
// Vector(int size, Item fillValue) -- vector w/ size entries all == fillValue
// int Length() -- returns size of vector (capacity)
// void SetSize(int newSize) -- resizes the vector to newSize elements
// void Fill(Item fillValue) -- set all entries == fillValue
// operator = -- assignment operator
// operator [] -- index operator

The List Class:

// List(); -- make an empty list
// void Append(const Type & t) -- add t to back of list
// void Prepend(const Type & t) -- add t to front of list
// void ChopFront() -- remove first element
// void Clear() -- make the list empty
// Type Front() -- return copy of first element in list
// Type Back() -- return copy of last element in list
// bool Contains(const Type & t) -- return true iff list contains t
// int Size(), int Length() -- return # elements in list
// void Print() -- print list elements
// First(), IsDone(), Next() -- iterate through all elements
// Type & Current() -- return element: can modify element
// Delete() -- delete current item
// InsertBefore(const Type & t) -- insert t before current item

The Matrix Class:

// Matrix() -- default, matrix of size 0x0
// Matrix(int rows, int cols) -- matrix with dimensions rows x cols
// Matrix(int rows, int cols, Item fillValue) -- matrix w/ entries all == fillValue
// int Rows() -- returns # of rows (capacity)
// int Cols() -- returns # of columns (capacity)
// void SetSize(int rows, int cols) -- resizes matrix to rows x cols
// void Fill(Item fillValue) -- set all entries == fillValue
// operator = -- assignment operator works properly
// operator [] -- indexes const and non-const matrices
Useful Functions

```c
int islower(int c)    // returns true when c is lowercase
int isupper(int c)    // returns true when c is upper case
int tolower(int c)    // returns lower-case equivalent of c;
                    // if !isupper(c) then returns c unchanged
int toupper(int c)    // returns upper-case equivalent of c;
                    // if !islower(c) then returns c unchanged
```

File Streams:

```c
string filename, word;
file = "testfile";
ifstream input(file);  // to declare an input file stream

while (input >> word)  // to use a file stream
{
    // Code
}
```

String Streams:

```c
string str, word;
str = "This is a test string"
istrstream readwords(str);     // binds str to the string stream

while (readwords >> word)  // to use a string stream
```

Getline:

```c
istream & getline(istream & is, string & s, char sentinel = '\n')
```