Answer all questions in a separate document and email that document to abrahamsonk@ecu.edu as an attachment. Make your answers clear, concise and precise.

Read each question twice. Be sure that you are answering the question that is asked. Check your answers.

The first two questions use the following structure type definition.

```c
struct Marx
{
    int Groucho;
    double Chico;

    Marx(int g, double c)
    {
        Groucho = g;
        Chico   = c;
    }
};
```

1. Suppose that you have already created a variable called \( m \) of type Marx. Which of the following statements will set the Groucho field of \( m \) to hold 24?

(a) \( m = \text{new Marx} (\text{Groucho} = 24) \);
(b) \( \text{Groucho} . m = 24 \);
(c) \( m . \text{Groucho} = 24 \);
(d) \( \text{*Marx} = 24 \);
(e) \( \text{Marx} m . \text{Groucho} = 24 \);

2. Which of the following statements will create a variable \( p \) of type Marx* and make \( p \) point to a new Marx whose Groucho variable holds 3 and whose Chico variable holds 9.0?

(a) \( \text{Marx*} p = \text{new Marx} (3, 9.0) \);
(b) \( \text{Marx*} p \{ \text{Groucho} = 3, \text{Chico} = 9.0 \} \);
(c) \( \text{Marx*} p = \text{new Marx*}; p->\text{Groucho} = 3; p->\text{Chico} = 9.0 \);
(d) \( \text{Marx*} p = \text{new Marx*}; \text{Groucho.p} = 3; \text{Chico.p} = 9.0 \);
(e) \( \text{new Marx*} p(3, 9.0) \);
Types ListCell and List are as defined in class. Here are their definitions.

```c
struct ListCell
{
    int       head;
    ListCell* tail;

    ListCell(int h, ListCell* t)
    {
        head = h;
        tail = t;
    }
};
typedef ListCell* List;
```

You can assume that constant emptyList and functions isEmpty(L), head(L), tail(L) and cons(x, L) have been defined as in class. They are summarized at the end of this test.

3. [MC] Which of the following will create variable L, of type List, and make it point to a new linked list holding 2, 4 and 6, in that order? (Using our conceptual list notation, it must make L refer to list [2, 4, 6].)

(a) List L = new List(2, 4, 6));
(b) List L = new ListCell(2, new ListCell(4, 6));
(c) List L = new ListCell(6, new ListCell(4, 2));
(d) List L = new ListCell(2, new ListCell(4, new ListCell(6, NULL))));
(e) List L = new ListCell(6, new ListCell(4, new ListCell(2, NULL))));

4. Suppose that variables L and n have already been created. L has type ListCell* and points to a linked list of length four, and n has type int. Which of the following sets variable n to the third integer in list L?

(a) n = L->head->tail->tail;
(b) n = L->tail->tail->head;
(c) n = L->tail->tail->tail;
(d) n = L->head->head->tail;
(e) n = L->tail->head->head;
5. Suppose that function foo(n) is already defined, with prototype

```c
int foo(int n);
```

You can use foo. Do not try to write a definition of foo.

Function mapfoo(L) takes a list L and returns a new list that has foo(n) at each position where L contains n. For example, using conceptual list notation, `mapfoo([3, 5, 7, 9]) = [foo(3), foo(5), foo(7), foo(9)]`.

(a) Write complete equations by filling in the blanks below to define `head(mapfoo(L))` and `tail(mapfoo(L))` for every nonempty list L. Use conceptual list notation, not C++ notation. Look at example

```
mapfoo([2, 4, 6, 8]) = [foo(2), foo(4), foo(6), foo(8)]
```

to help you decide what to write. If your answers are not equations, they will receive a score of 0.

```latex
\begin{align*}
\text{head}(&\text{mapfoo}(L)) = \\
\text{tail}(&\text{mapfoo}(L)) =
\end{align*}
```

(b) Write complete equations by filling in the blanks below so that, taken together, your equations define `mapfoo(L)` for every list L. Use conceptual list notation, not C++ notation. Use your equations from part (a) to help you write the second equation below. If your answers are not equations, they will receive a score of 0.

```latex
\begin{align*}
\text{mapfoo}(&[]) = \\
\text{mapfoo}(&L) = \\
\text{(when } &L \neq [])
\end{align*}
```

(c) Write a C++ definition of function `mapfoo(L)`. Function `mapfoo(L)` must not modify list L. It must create a new list. Do not use any kind of loop. A heading is given.

```c
ListCell* mapfoo(const ListCell* L)
```
Summary of conceptual list notation.

- `[]` is an empty list
- `isEmpty([])` is true
- `head([2, 4, 6, 8]) = 2`
- `tail([2, 4, 6, 8]) = [4, 6, 8]`
- `2 : [4, 6, 8] = [2, 4, 6, 8]`

(h : t is the list whose head is h and whose tail is t.)

Summary of C++ definitions that you can use that are related to conceptual list notation.

<table>
<thead>
<tr>
<th>C++</th>
<th>Conceptual</th>
</tr>
</thead>
<tbody>
<tr>
<td>emptyList</td>
<td>[]</td>
</tr>
<tr>
<td>isEmpty(L)</td>
<td>isEmpty(L)</td>
</tr>
<tr>
<td>head(L)</td>
<td>head(L)</td>
</tr>
<tr>
<td>tail(L)</td>
<td>tail(L)</td>
</tr>
<tr>
<td>cons(x, L)</td>
<td>x : L</td>
</tr>
</tbody>
</table>