Computer Science 2530
Spring 2020
Practice Exam 4 Answers

Answers to the multiple choice questions are in bold.
The first two questions use the following structure type definition.

```
struct Feline
{
  int         size;
  const char* coat;
  Feline(int s, const char* c)
  {
    size  = s;
    coat  = c;
  }
};
```

1. [MC] Suppose that you have already created a variable called cat of type Feline. Which of the following statements will set the 'size' field of cat to hold 10?

   (a) s.Feline = 10;

   (b) cat.s = 10;

   (c) **cat.size = 10;**

   (d) Feline.size = 10;

   (e) size.cat = 10;

2. [MC] Which of the following statements or sequences of statements will create a variable p of type Feline* and make p point to a new Feline structure whose 'size' field holds 8 and whose 'coat' field holds "tabby"?

   (a) new Feline* p(8, "tabby");

   (b) Feline* p = new Feline*; p-¿size = 8; p-¿coat = "tabby";

   (c) **Feline* p = new Feline(8, "tabby");**

   (d) Feline* p(8, "tabby");

   (e) Feline* p = new Feline*; size.p = 8; coat.p = "tabby";

Types ListCell and List are as defined in class. Here are their definitions.

```
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.

3. [MC] Which of the following will create variable L, of type List, and make it point to a new linked list holding 4 and 7, in that order? (Using our conceptual list notation, it must make L refer to list [4, 7].)

   (a) **List L = new ListCell(4, new ListCell(7, NULL));**

   (b) List L = new ListCell(7, new ListCell(4, NULL));

   (c) List L = new List(4, new List(7, NULL));

   (d) List L = new ListCell(4, 7);

   (e) List L = new ListCell(7, 5);

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

   (a) n = L->head->tail

   (b) n = L->tail->tail;

   (c) **n = L->tail->head;**

   (d) n = L->2;

   (e) n = L[1];

5. Suppose that sum(L) is intended to return the sum of the values in list L. For example, sum([8, 2, 5]) = 8 + 2 + 5 = 15 and sum([9, 7]) = 9 + 7 = 16. The sum of an empty list is 0.

   (a) ***Using the conceptual notation for lists discussed in class***, complete the following equations so that, taken together, they define sum(L) for every list L. Use conceptual notation, not C++ notation, for this part. See the bottome of the last page for a brief summary of conceptual list notation.

$$\begin{aligned} \text{sum}([\,]) &= \mathbf{0} \\ \\ \text{sum}(L) &= \mathbf{head(L)} + \mathbf{sum(tail(L))} \\ &\quad (\text{when } L \neq [\,]) \end{aligned}$$

   (b) Following your equations from part (a) closely, write a C++ definition of sum(L). It must not change any of the cells in list L. **Do not use any kind of loop for this definition.** A heading is given.

   Using the C++ version of conceptual notation:

```
int sum(List L)
{
  if(isEmpty(L))
  {
    return 0;
  }
  else
  {
    return head(L) + sum(tail(L));
  }
}
```

   Using native C++ notation:

```
int sum(List L)
{
  if(L == NULL)
  {
    return 0;
  }
  else
  {
```

```
    return L->head + sum(L->tail);
  }
}
```

6. Suppose that function negatives(L) is intended to be a nondestructive function that returns a list obtained from list L by replacing each value x by $-x$. For example,

   - negatives([3, 0, −9]) = [−3, 0, 9],
   - negatives([−3, 0, 9]) = [3, 0, −9],
   - negatives([4]) = [−4],
   - negatives([−4]) = [4],
   - negatives([ ]) = [ ].

   (a) ***Using the conceptual notation for lists discussed in class***, complete the following equations so that, taken together, they define negatives(L) for every list L. Use the examples above to help you work these out. **Do not guess. Do not use C++ notation.**

$$\begin{aligned} \text{negatives}([\,]) &= [\,] \\ \text{negatives}(L) &= -\mathbf{head(L)} : \mathbf{negatives(tail(L))} \\ &\quad (\text{when } L \neq [\,]) \end{aligned}$$

   (b) Demonstrate that your equations are correct by using them to compute, in order,

   1. negatives([ ]) = [ ]

   2. $$\begin{aligned} \text{negatives}([5]) &= \mathbf{-head([5]) : negatives(tail([5]))} \\ &= \mathbf{-5 : negatives([\,])} \\ &= \mathbf{-5 : [\,]} \\ &= \mathbf{[-5]} \end{aligned}$$

3. $\text{negatives}([3,5])$ $=$ **-head([3, 5]) : negatives(tail([3, 5]))**
$=$ **-3 : negatives([5])**
$=$ **-3 : [-5]**
$=$ **[-3, -5]**

Do not just write the answers without using your equations. If you discover that your equations are not correct then fix them and recompute each of the above.

(c) Following your equations from part (a) closely, write a C++ definition of negatives(L). **It must not change any of the cells in list L. Do not use any kind of loop for this definition.** A heading is given.

Using the C++ version of conceptual notation:

```
List negatives(List L)
{
  if(isEmpty(L))
  {
    return emptyList;
  }
  else
  {
    return cons(-head(L), negatives(tail(L)));
  }
}
```

Using native C++ notation:

```
List negatives(List L)
{
  if(L == NULL)
  {
    return NULL;
  }
  else
  {
    return new ListCell(-(L->head), negatives(L->tail));
  }
}
```

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]