ECE 218
Second Test
10th November 2021

Who are you?

_____

What is your C number?

_____

Sign the honour pledge:

_____

Do not write in these boxes

Question	Out of	Your score
5	33	
6	33	
7	33	
8	1	

Question 5: Vectors

You are going to be storing information on a lot of items sold by a supermarket.
You are to store them in a vector that you code yourself.
Do not #include <vector>, define the vector class yourself.

For each item, the following pieces of information will be stored:
description, number in stock, price each, and UPC (bar code).

A. Define a struct, with a constructor and print method, suitable for representing an item.

B. Define a class, using public and protected in an appropriate way, to represent a vector of items. It should have a constructor and a method for adding a new item.

C. Write a method that returns as its result the total value of all items in the supermarket.

A.

```
struct item
{ string desc, upc;
  int stock, price;

  item(string d, int s, int p, string u)
  { desc = d; stock = s; price = p; upc = u; }

  void print()
  { cout << desc << ", " << stock << " in stock, "
         << price << " cents each, " << upc << "\n"; } };
```

B.

```
class vec
{ protected:
    item * * data;
    int num, cap;

  public:
    vec()
    { data = NULL;
      num = 0;
      cap = 0; }

    growto(int newcap)
    { item * * newdata = new item * [newcap];
      for (int i = 0; i < num; i += 1)
        newdata[i] = data[i];
```

```
      delete [] data;
      data = newdata;
      cap = newcap; }

    void add(item * i)
    { if (num == cap)
        growto(cap==0 ? 1 : cap * 2);
      data[num] = i;
      num += 1; } };
```

C, inside the public section of the class vec

```
    int totalvalue()
    { int t = 0;
      for (int i = 0; i < num; i += 1)
        t += data[i]->price * data[i]->stock;
      return t; }
```

Question 6: Fast Sorting

A person is to be represented by a struct containing their name, address, and phone number.
The array A contains N pointers to person objects.

Give C++ code that would sort that array (so that their names are in alphabetical order) in time proportional to N*log2(N)

```
void qs(person * * A, int first, int last)
{ if (last-first > 0)
    return;
  int pp = random() % (last - first + 1) + first;
  person * p = A[pp];
  swap(A[pp], A[last]);
  int i = 0, s = 0;
  while (i < last)
  { if (A[i]->name < p->name)
    { swap(A[i], A[s]);
      s += 1; }
    i += 1; }
  swap(A[s], A[last]);
  qs(A, first, s - 1);
  qs(A, s + 1, last); }
```

Question 7: Linked Lists

A person is to be represented by a struct containing their name, address, and phone number.
This question is about a linked list of persons.

A. Define the structs or classes required to represent a linked list of persons, give each a constructor.

B. Write a function or method that adds a person to the front of an existing linked list.

C. Write a function or method that adds a person to the end of an existing linked list.

D. Write a function or method that finds and removes the person from the linked list whose name comes first in alphabetical ordering.

E. Use your existing work to write a function or method that sorts a whole linked list of persons.

A.

```
struct person
{ string name, addr, ph; };

struct link
{ person * data;
  link * next;
  link(person * p, link * n)
  { data = p; next = n; } };

class list
{ protected:
    link * head;
  public:
    list()
    { head = NULL; }
```

B.

```
    void add_front(person * p)
    { head = new link(p, head); }
```

C. The question didn't say it had to be efficient, watch out for that.

```
    void add_end(person * p)
    { if (head == NULL)
      { head = new link(p, NULL);
        return; }
      link * ptr = head;
```

```
  while (ptr->next != NULL)
    ptr = ptr->next;
  ptr->next = new link(p, NULL); }
```

D.

```
person * remove_little()
{ assert(head != NULL)
  link * prev = NULL;
  link * small = head;
  link * prevtosmall = NULL;
  link * ptr = head->next;
  while (ptr != NULL)
  { if (ptr->data->name < small->data->name)
    { small = ptr;
      prevtosmall = prev; }
    prev = ptr;
    ptr = ptr->next; }
  if (prevtosmall == NULL)
    head = small->next;
  else
    prevtosmall->next = small->next;
  person * result = small->data;
  delete small;
  return result; }
```

E.

```
void sort()
{ list R;
  while (head != NULL)
  { person * p = remove_little();
    R.add_end(p); }
  head = R.head; }
```

Question 8: Multivariate Calculus

Draw a picture of a cat sitting on a chair.