Midterm Answer Key

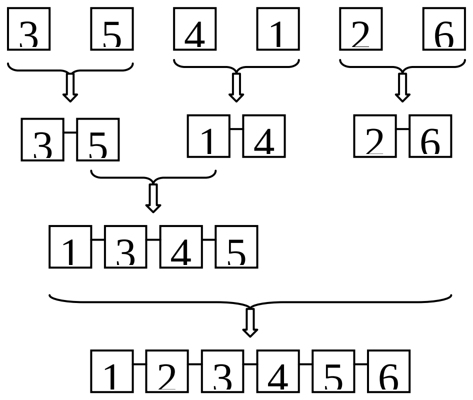
1. a.)

a.)	b.)
4	2
6	3
12	4
15	9
16	1
24	3
	4

2.
```
// allocate nrow x ncol 2d array
arr2d = (int**)malloc(nrow*sizeof(int*));
for (i = 0; i < nrow; i++) {
   arr2d[i] = (int*)malloc(ncol*sizeof(int));
}

// free array storage
for (i = 0; i < nrow; i++) {
   free(arr2d[i]);
}
free(arr2d)
```

3.
```
class Pet {
public:
      // constructors & destructors
      Pet();
      Pet(char *n, int a, float p, char s);
      ~Pet();

      // accessors
      char *getName();
      int getAge();
      float getPrice();
      char getSex();

      // manipulators
      void setName(char *n);
      void setAge(int a);
      void setPrice(float p);
      void setSex(char s);

      // other methods
      void read();
      void print();
      void incrementAge();

private:
      char name[50];
      int age;
      float price;
      char sex;
};

float Pet::getPrice() {
      return price;
}

void Pet::setPrice(float p) {
      price = p;
}

Pet::Pet() {
      strcpy(name,"");
      age = 0;
      price = 0;
      sex = '\0';
}

Pet::Pet(char *n, int a, float p, char s) {
      strcpy(name,n);
      age = a;
      price = p;
      sex = s;
}

void Pet::incrementAge() {
      age++;
}
```

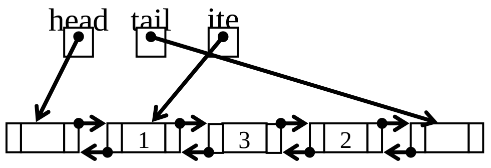
4.



5.

```
void insertion_sort(int *arr, int arr_len) {
        int i, j;
        int tmp;

        // loop through entire list
        for (i = 0; i < arr_len; i++) {
                // insert item in proper sorted location
                for (j = i; j > 0; j--) {
                        // see if we need to move current item
                        // to a lower position
                        if (arr[j] < arr[j-1]) {
                                // swap items
                                tmp = arr[j];
                                arr[j] = arr[j-1];
                                arr[j-1] = tmp;
                        }
                }
        }
}
```

6.



7.

```
IntLinkItem* findMinNode(IntList list) {
        IntLinkItem *item, *min;

        min = list.getHead()->getNext();
        for (item = list.getHead()->getNext(); item != list.getTail();
item = item->getNext()) {
                if (item->getData() < min->getData()) {
                        min = item;
                }
        }
        return min;
}
```

8.

```
void cycleList(IntList list, int ncycle) {
        IntLinkItem *first;
        int i;

        for (i = 0; i < ncycle; i++) {
                first = list.getHead()->getNext();
                if (first != list.getTail()) {
                        first->excise();
                        first->insertBefore(list.getTail());
                }
        }
}
```