COMP 2401 B

Test #2 (version 3)

1. [2 marks] a
2. [2 marks] c
3. [2 marks] d
4. [2 marks] c
5. [2 marks] a
6. [2 marks] a (alt: d)

7. [10 marks]

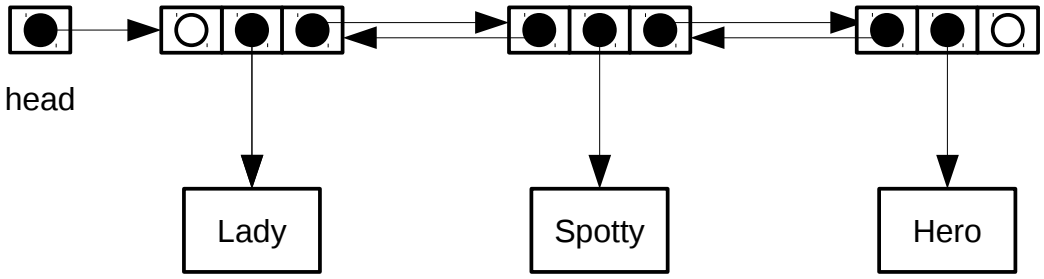
```
void initAcct(char *n, int i, AcctType **acct) {
  *acct = malloc(sizeof(AcctType));
  strcpy((*acct)->name, n);
  (*acct)->id = i;
}

int main()
{
  AcctType *newAcct;
  initAcct("Gertrude", 3554, & newAcct);
  printf("Name is %s, id is %d\n", newAcct->name, newAcct->id);
  free(newAcct);
}
```

Marking:
-- 2 marks for making parameter a double pointer in `initAcct()`
-- 2 marks for allocating `AcctType` in `initAcct()`
-- 2 marks for dereferencing `acct` in `initAcct()` (1 mark each)
-- 2 marks for passing address of `newAcct` to `initAcct()`
-- 2 marks for freeing `newAcct`

8. [28 marks]

a. [4 marks]



Marking:
-- 1 mark for correct pointer to head node
-- 1 mark for 2 next pointers
-- 1 mark for 2 prev pointers
-- 1 mark for correct pointers to data structures, in correct order

b. [12 marks]

```
  NodeType *newNode;
  NodeType *currNode;
  NodeType *lastNode;

// 4 marks for allocating and initializing node
// -- 2 marks for malloc (zero if freed)
// -- 2 marks for initializing node data, next and prev
  newNode = (NodeType *) malloc(sizeof(NodeType));
  newNode->data = newRunner;
  newNode->prev = NULL;
  newNode->next = NULL;

// 2 marks for dealing with empty list case
  if (list->head == NULL) {
    list->head = newNode;
    return;
  }

// 2 marks for correctly looping through list and saving last node
  currNode = list->head;
  lastNode = NULL;
  while (currNode != NULL) {
    lastNode = currNode;
    currNode = currNode->next;
  }
// 2 marks for setting last node's next to new node
  lastNode->next = newNode;

// 2 marks for setting new node's prev to last node
  newNode->prev  = lastNode;
```

c. [12 marks]

```
  NodeType *newHead;
  RunnerType *goner;

// 2 marks for dealing with empty list case
  if (list->head == NULL)
    return 0;

// 2 marks for saving new head
  newHead = list->head->next;

// 1 mark for saving current head's data
  goner = list->head->data;

// 2 marks for freeing current head node
  free(list->head);

// 2 marks for setting new head
  list->head = newHead;

// 2 marks for setting new head's prev to NULL
  newHead->prev = NULL;

// 1 mark for returning last node's data
  return goner;
```