2 Programming in C (NK)

Consider the following structure declaration for a linked list in C:

```
struct node {
  int data;
  struct node* tail;
};
typedef struct node Node;
```

We represent linked lists as pointers to `Node` structs. Empty lists are represented with the null pointer. Non-empty lists are represented as a valid pointer to a `Node`, with the head data in the `data` field of the struct and the tail in the `tail` field of the struct. All lists will be assumed to be non-cyclic.

(*a*) Write a function to add an element to the head of the list with the following prototype.

```
Node *cons(int n, Node *tail);
```

[5 marks]

(*b*) Write a function to free the memory associated with a linked list, with the following prototype:

```
void free_list(Node *list);
```

[5 marks]

(*c*) Write a function to do an in-place list reversal. It should take a list as an argument, and return a list with the elements reversed. This function may not do any heap allocation, but may modify its input. It should have the following prototype:

```
Node *reverse(Node *list);
```

[10 marks]