Lab 9: Recursion, Binary Search Trees, & Linked Lists

Aim

This lab class comprises pen-and-paper tasks. It will provide you with an opportunity to:

- explore algorithmic recursion — which is important in the processing of self-referential data structures;
- practise manipulating binary search trees; and
- write code to continue to develop your linked-list programming skills.

Tasks

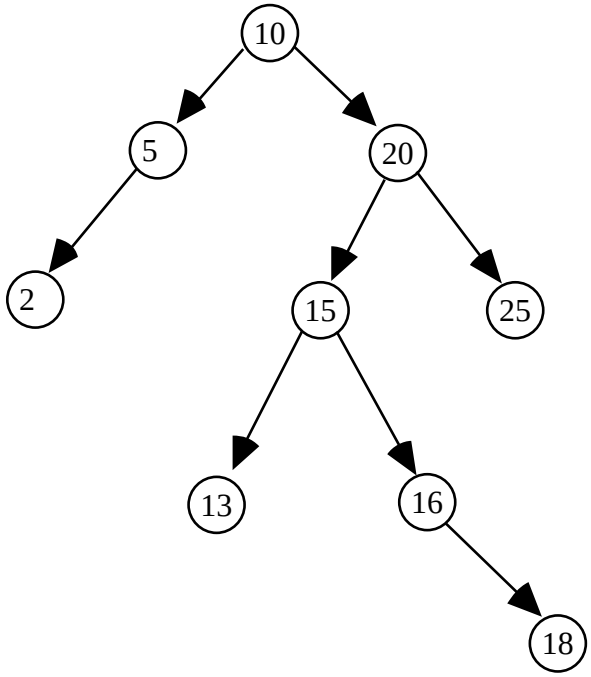
1. Rabbits are very prolific breeders. If rabbits did not die, their population would very quickly get out of hand. The following are some statistics obtained in a recent survey of randomly selected rabbits:
   - rabbits never die;
   - a rabbit reaches sexual maturity exactly two months after birth (i.e. at the beginning of its third month of life); and
   - rabbits are always born in male-female pairs. At the beginning of every month, each sexually mature male-female pair gives birth to exactly one male-female pair.

   (i) Suppose (at the beginning of month 1) that we started with one newborn male-female pair. Using pen and paper, how many pairs would there be in the sixth month, counting the births that took place at the beginning of month 6?

   (ii) Construct a recursive expression that can be used for computing the value of `Rabbit(n)`, which should be the number of pairs alive in month `n` in terms of the population in early months. *Hint*: start by considering how the number of rabbits alive in the requested month is related to the number alive in the previous month.

   (iii) Consider the terminating case(s) for this scenario. Is there one terminating case for this recursive definition? If so what is it? If not, how many are there and what are they? *Note*: time starts at month 1.

2. Consider the following binary search tree:



(i) Add value ‘12’ to the tree.

(ii) Add value ‘17’ to the original tree.

(iii) Delete value ‘5’ from the original tree.

(iv) Delete value ‘10’ from the original tree.

3. A linked list of integers may be implemented in C using the following types:

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>

typedef struct node_int {
     int data;
     node next;
} *node;

void init_node(node *n, int o)
{
     *n = (node)malloc(sizeof(struct node_int));
     (*n)->data = o;
     (*n)->next = NULL;
}

int get_data(node n)
{
     return (n->data);
}

node get_next(node n)
{
     return (n->next);
}

void set_data(node n, int o)
{
     n->data = o;
}

void set_next(node v, node n)
{
     v->next = n;
}


typedef struct list_int {
     node front;
} *list;
```

Write a C function to delete the largest element in a list — you may assume that the list is not empty and that the largest element value occurs only once. Can this be done with a single traversal of the list?