Full code

Pythontutor

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

typedef struct Person {
  char name[10];
  int age;
  struct Person* friends[3];
} Person;

typedef struct Node {
  Person data;
  struct Node* next;
} Node;

void print_list(Node* list) {
  while (list != NULL) {
    printf("%s\t%d\n", list->data.name, list->data.age);
    list = list->next;
  }
}

int size(Node* list) {
 // return number of elements in the list
// int s = 0;
//   while (list != NULL) {
//     s++;
//     list = list->next;
//   }
// return s;
  return list == NULL? 0: 1+ size(list->next);
}

Person* element_at(Node* list, int pos) {
  // return the element at the 'pos' position of the linked list
  int s = 0;
  while (list != NULL) {
    if (s == pos) return &(list->data);
    s++;
    list = list->next;
  }
  return NULL;
  // return pos == 0? &(list->data) : element_at(list->next, pos-1);
}

Node* append(Node* list, Person* data) {
  // add new person data as the last element in the list
  // and return the pointer to the first element in the list.

  Node* new_element = malloc(sizeof(Node));
  new_element->data = *data;
  new_element->next = NULL;

  if (list != NULL) {
    Node* head = list;
    while (list->next != NULL) {
      list = list->next;
    }
    list->next = new_element;
    return head;
  } else return new_element;

}


int main() {

  Node third = {
    {"Alice", 22},
    NULL
  };

  Node second = {
    {"Bob", 26},
    &third
  };

  Node first = {






                 );
                 : is %d.\n", s);
                 s; i++) {
                 _at(&first, i);
                 ent is %s.\n", i, s->data.name);

  Person new_person = { "Diestel", 27 };
  Node* list = append(&first, &new_person);
  print_list(&first);
  s = size(&first);
  printf("size of list is %d.\n", s);
```

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

HW: Insert element at a position in the list and return pointer to first element

```
Node* insert(Person p, int pos, Node* l) {
    // TODO
}
```

HW: Concatenate 2 lists and return pointer to first element

```
Node* concat(Node* l1, Node* l2) {
    // TODO
}
```

HW: Reverse a list and return pointer to first element

```
Node* reverse(Node* l) {
    // TODO
}
```

Social Network

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

```
typedef struct Person {
    char name[10];
    int age;
    struct Person* friends[5];
    int num_friends;
} Person;

typedef struct SociaslNet {
    Person members[100];
    int size;
} SocialNet;
```

```c
void print_person(Person* p) {
        printf("%s\t%d\t%s\t\t\t", p->name, p->age);
        for (int i =0; i < p->friends_size; i++) {
                printf("%s,", (p->friends[i])->name);
        }
        printf("\n");
}

void print_socialnet(SocialNet *s) {
        printf("------------------------------\n");
        printf("Name\tAge\tFriends\n");
        printf("------------------------------\n");
        for (int i = 0; i < s->size; i++ ) {
                print_person(&(s->members[i]));
        }
        printf("------------------------------\n");
}

void add_friend(Person* p, Person* f) {
        p->friends[p->friends_size] = f;
        p->friends_size += 1;
}

int main() {
        SocialNet s = {
                .members = {
                        { "Ramu", 19, .friends_size = 0},
                        { "Ammu", 21, .friends_size = 0},
                        { "Vinod", 24, .friends_size = 0}
                },
                .size = 3
        };
        add_friend(&(s.members[0]) , &(s.members[1]) );
        add_friend(&(s.members[0]) , &(s.members[2]) );
        add_friend(&(s.members[1]) , &(s.members[2]) );
        add_friend(&(s.members[2]) , &(s.members[0]) );

        print_socialnet(&s);


        return 0;
}
```

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

HW: Finding a person by name

```
Person* find_person(char* name1, SocialNet *sn) {
    // TODO (solution at the end of page)
}
```

INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY
HYDERABAD

HW: Check Mutual Friends by name

```c
bool check_mutual_friends(char *name1, char *name2, SocialNet *sn) {
 // TODO p and q are mutual friends if q is in the friend list of p
 // and p is in the friend list of q
}
```

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD