LECTURE C PROGRAMMING POINTERS AND STRUCTURES

Computer Systems and Networks

Dr. Pallipuram
(vpallipuramkrishnamani@pacific.edu)

University of the Pacific

Today's Class

- Pointers and data structures
  - Structures
  - linked lists

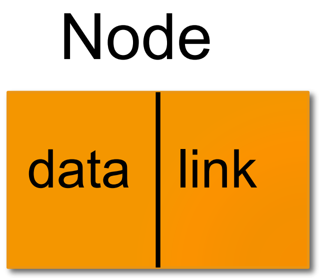
C Structures

Structures are a nice way to bring certain related items together

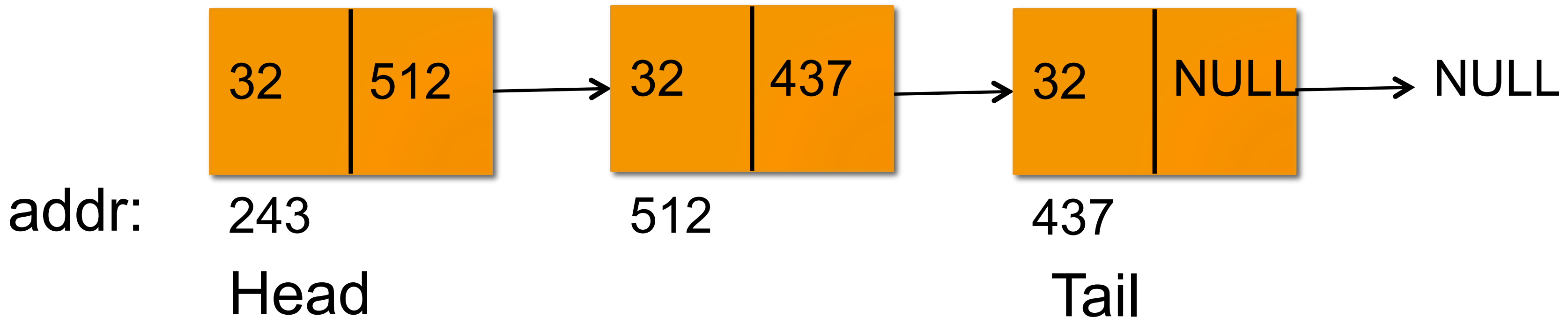
```
struct database
{
  int id_number;
  int age;
  float salary;
};
int main()
{
  struct database employee; //an object
  employee.age = 22;
  employee.id_number = 1;
  employee.salary = 12000.21;
}
```

Pointers and Structures

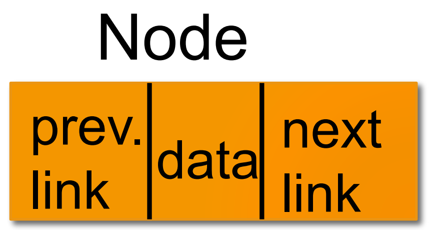
Singly Linked Lists are a bunch of dynamically allocated structures (Nodes) connected to each other

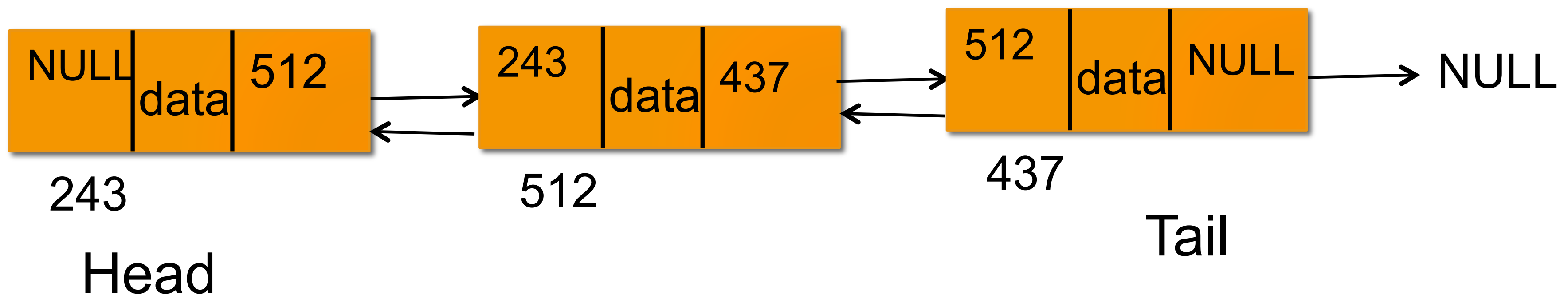


link stores the address of the next node



Doubly Linked Lists





Example – Creating a doubly linked list for the data: struct coordinate

Discuss a function called: `create_list` to create a doubly linked list for the structure, `coordinate`, containing `height` and `width` information for a pixel. This function should be called to add a new coordinate to the list. The list should be doubly linked. Additionally, the list should have a `head` (first node) and a `tail` (last node) for easy traversal.

```
struct double_list {
struct coordinate coord;
struct double_list *next,*prev;
};
```

```
struct coordinate {
int height,width;
};
```

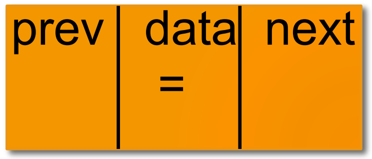
```
struct double_list {
struct coordinate coord;
struct double_list *next,*prev;};

main() {
//Something something
struct coordinate seedpoint;
struct double_list *head,*tail;
head=NULL; tail=NULL;
```

No List yet

```
struct double_list {
struct coordinate coord;
struct double_list *next,*prev;}

main() {
//Something something
struct coordinate seedpoint;
struct double_list *head,*tail;
head=NULL; tail=NULL;

//something something
//a loop to push random seedpoints to list 3 times
for(i=0;i<3;i++) {
   //something something creates seedpoints

   create_list(&head, &tail, seedpoint);
}
```

```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }

}
```

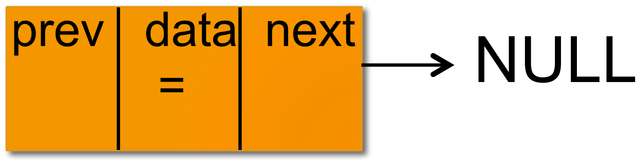
Let’s execute first three iterations



```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
 →    (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
```

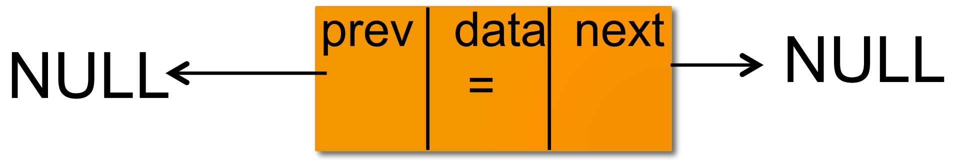
iteration 0 – nothing yet

```
}
```



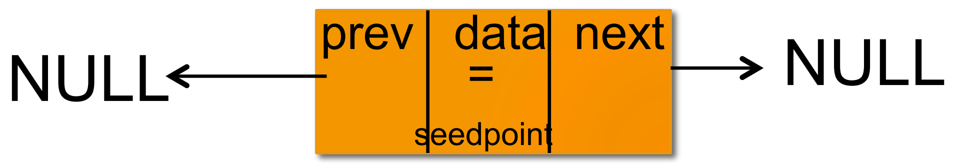
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
  →   (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

iteration 0 – nothing yet



```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
 -->  (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

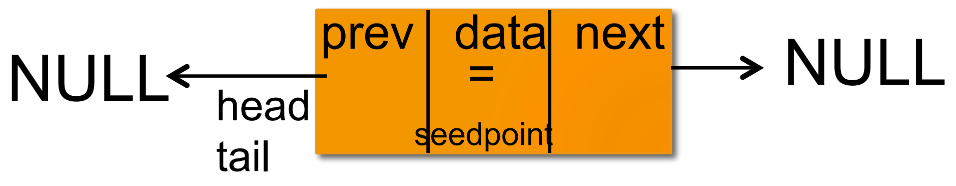
iteration 0 – nothing yet



```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
 →    (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
```

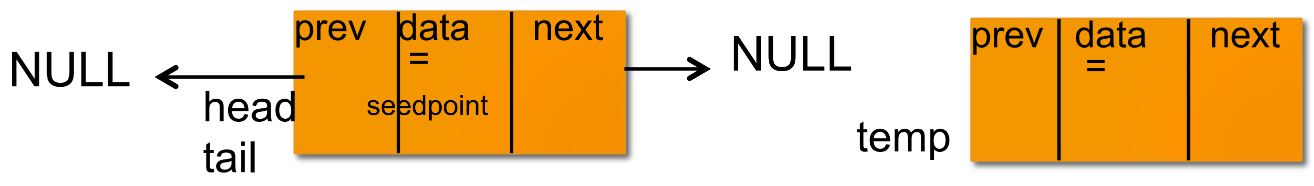
iteration 0 – nothing yet

```
}
```



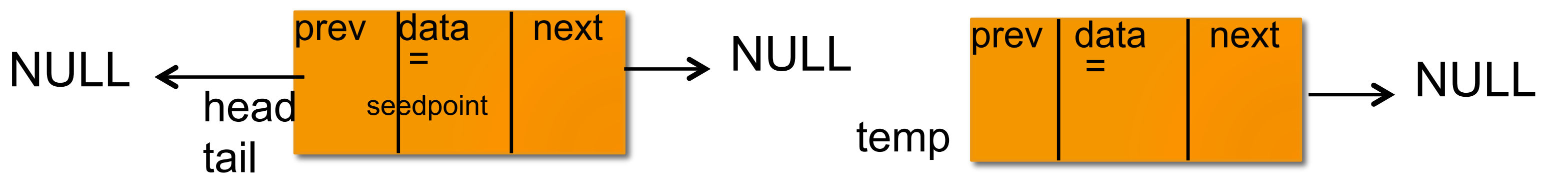
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
  →   (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
   }
}
```

iteration 0 – nothing yet



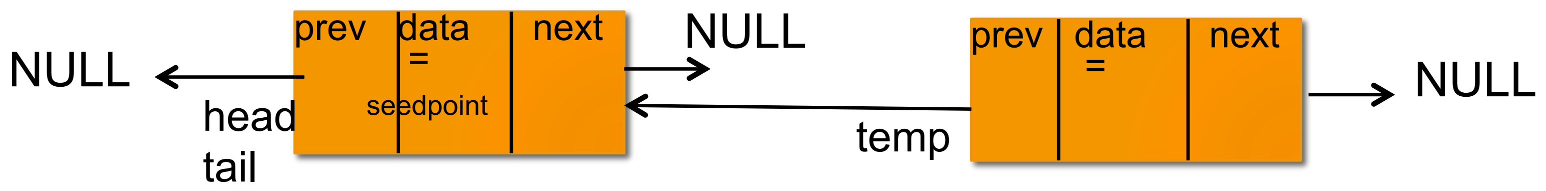
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
→     temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

iteration 1 – one item on the list



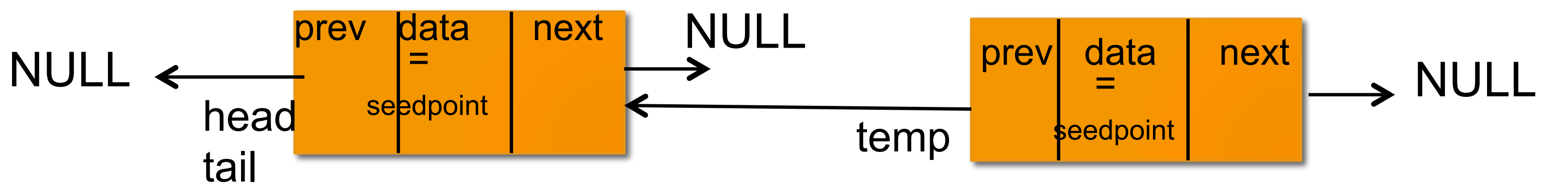
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
  →   temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

iteration 1 – one item on the list



```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
 →    temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

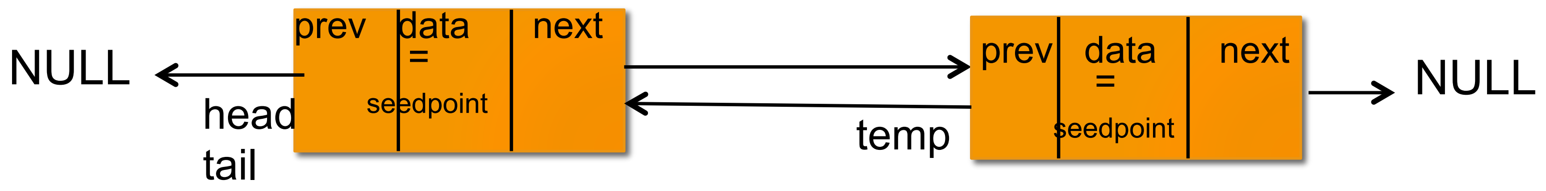
iteration 1 – one item on the list



```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
  →   temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
```

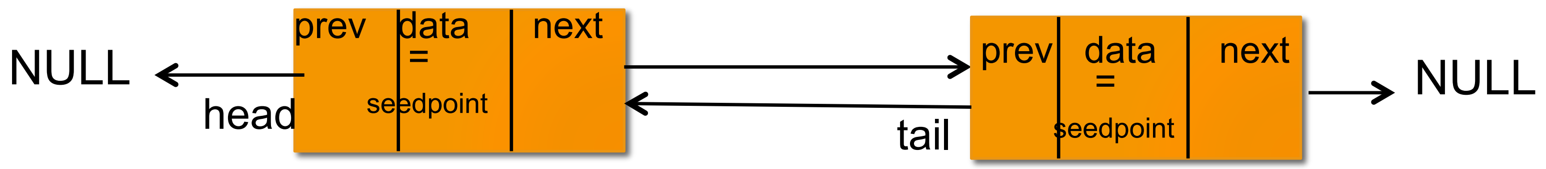
iteration 1 – one item on the list

```
}
```



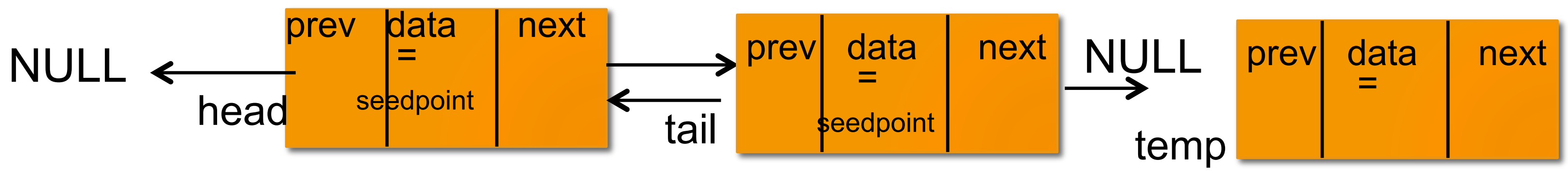
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
  →   (*tail)->next = temp;
      (*tail)=temp;
   }
}
```

iteration 1 – one item on the list



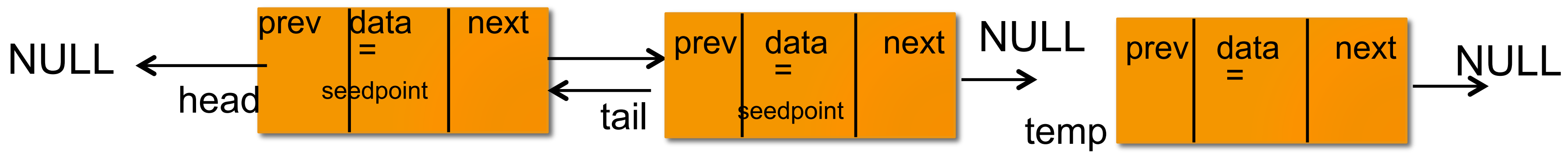
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
  →   (*tail)=temp;
   }
}
```

iteration 1 – one item on the list



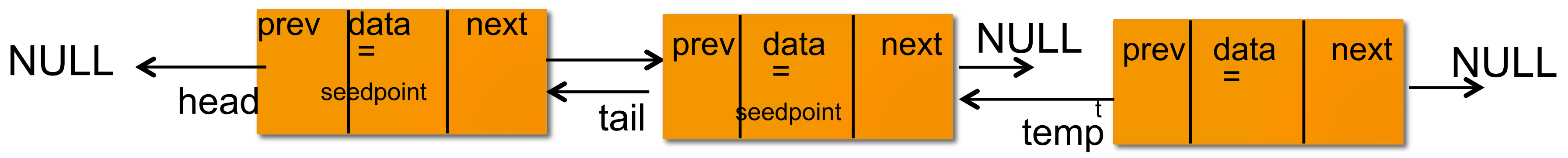
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

iteration 2 – two items on the list



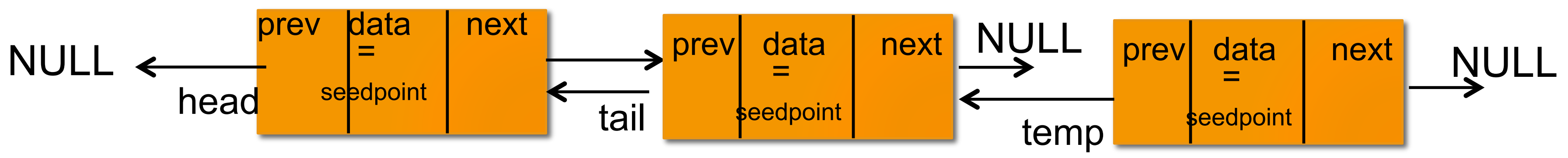
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
 ⟶   temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

iteration 2 – two items on the list



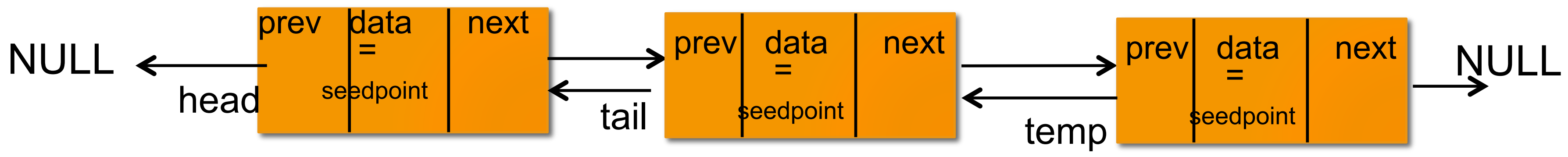
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
  →   temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

iteration 2 – two items on the list



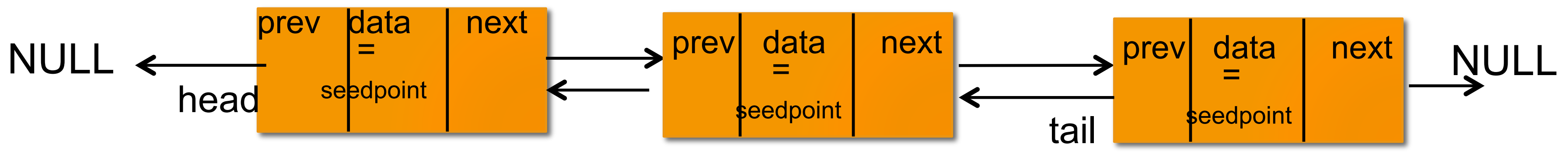
```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
  →   temp->coord=seedpoint;
      (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

iteration 2 – two items on the list



```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
  →   (*tail)->next = temp;
      (*tail)=temp;
  }
}
```

iteration 2 – two items on the list



```
void create_list(struct double_list **head, struct double_list **tail,
struct coordinate seedpoint) //call by reference
{
   struct double_list *temp;

   if(*head==NULL) //nothing on the list yet
   {
      (*head)=(struct double_list *)malloc(sizeof(struct double_list));
      (*head)->next=NULL;
      (*head)->prev=NULL;
      (*head)->coord=seedpoint;
      (*tail)=(*head); //tail and head are same when only 1 item
   }
   else {
      temp = (struct double_list *)malloc(sizeof(struct double_list));
      temp->next = NULL;
      temp->previous = *tail;
      temp->coord=seedpoint;
      (*tail)->next = temp;
 -->  (*tail)=temp;
  }
```

iteration 2 – two items on the list

```
}
```

and so on!

Discuss a C function that traverses the doubly linked list you just created and prints the data. Start the traversal from the head.

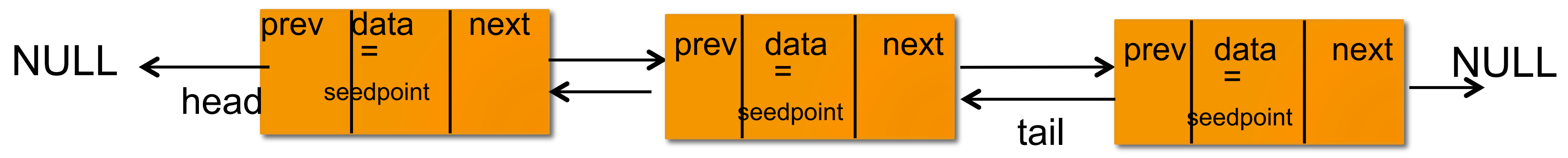
SOLUTION

```
void print_list(struct double_list *head) //call
by value. No change to list
{
   struct double_list *temp;
   temp=head;

   while(temp!=NULL)
   {
      printf("\n %d %d",(temp->coord).height,
(temp->coord).width);
      temp=temp->next;
   }
}
```

Discuss

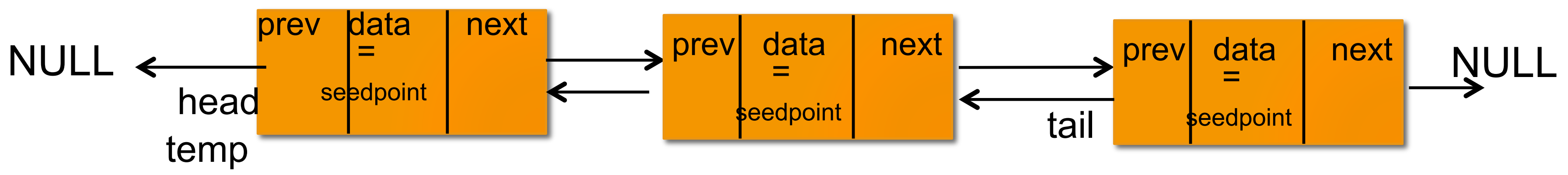
Let us simulate a queue (FIFO list). The linked list function we wrote already creates a queue. Write a C function that returns the first element (`struct coordinate` type), removes the node, and adjusts the linked list.

```
struct coordinate exit_queue(struct double_list **head) //
call by reference
{  struct coordinate seedpoint;
   struct double_list *temp;
   temp=*head;

   if(temp==NULL) //nothing on the list yet
   {
      printf("\n Nothing to exit..");
      exit(0);
   }
   else {
          seedpoint=temp->coord;
          (*head)=(*head)->next;
          free(temp);
           if((*head)!=NULL)
              (*head)->prev=NULL;
      return seedpoint;
  }
}
```



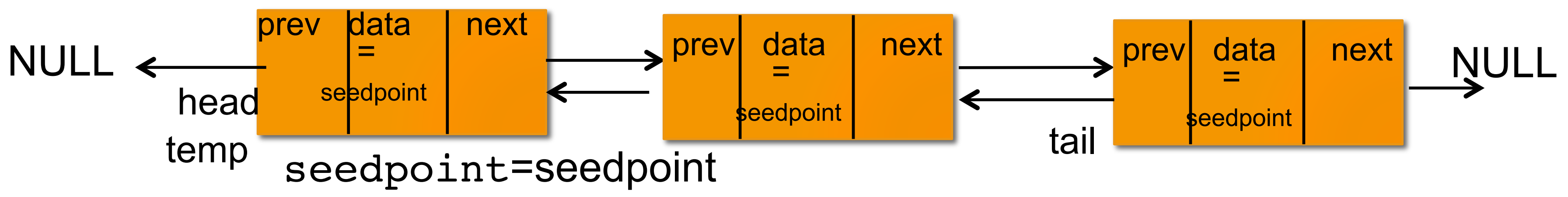
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
              seedpoint=temp->coord;
              (*head)=(*head)->next;
              free(temp);
               if((*head)!=NULL)
                  (*head)->prev=NULL;
       return seedpoint;
   }
}
```

Let's call this function three times



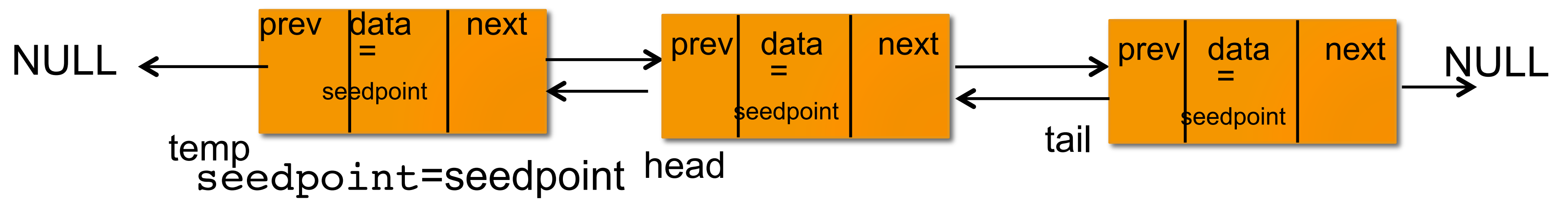
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{  struct coordinate seedpoint;
   struct double_list *temp;
   temp=*head;

   if(temp==NULL) //nothing on the list yet
   {
      printf("\n Nothing to exit..");
      exit(0);
   }
   else {
          seedpoint=temp->coord;
          (*head)=(*head)->next;
          free(temp);
           if((*head)!=NULL)
              (*head)->prev=NULL;
      return seedpoint;
  }
}
```

first call



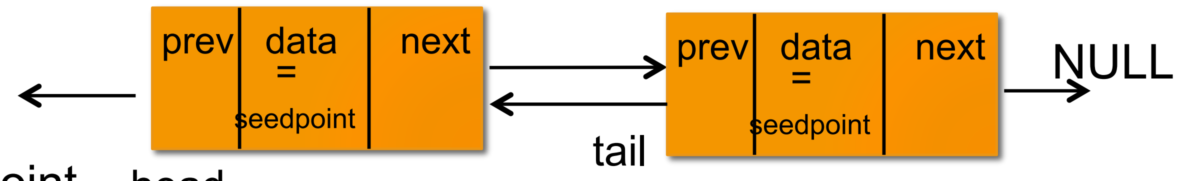
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
        printf("\n Nothing to exit..");
        exit(0);
    }
    else {
    --->    seedpoint=temp->coord;
            (*head)=(*head)->next;
            free(temp);
             if((*head)!=NULL)
                 (*head)->prev=NULL;
        return seedpoint;
   }
}
```

first call



```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{  struct coordinate seedpoint;
   struct double_list *temp;
   temp=*head;

   if(temp==NULL) //nothing on the list yet
   {
      printf("\n Nothing to exit..");
      exit(0);
   }
   else {
          seedpoint=temp->coord;
          (*head)=(*head)->next;
          free(temp);
           if((*head)!=NULL)
              (*head)->prev=NULL;
      return seedpoint;
  }
}
```

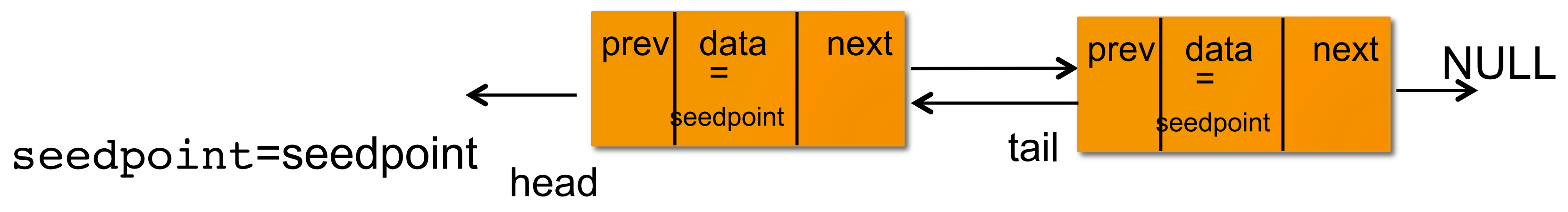
first call



seedpoint=seedpoint head

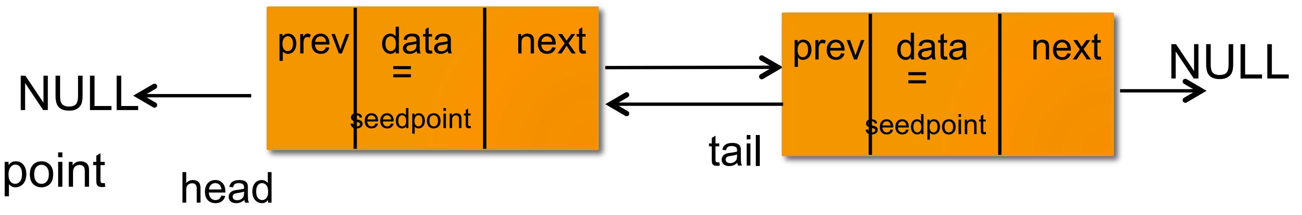
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
           seedpoint=temp->coord;
           (*head)=(*head)->next;
      -->  free(temp);
            if((*head)!=NULL)
                (*head)->prev=NULL;
       return seedpoint;
   }
}
```

first call



```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
              seedpoint=temp->coord;
              (*head)=(*head)->next;
              free(temp);
      →        if((*head)!=NULL)
                   (*head)->prev=NULL;
       return seedpoint;
   }
}
```

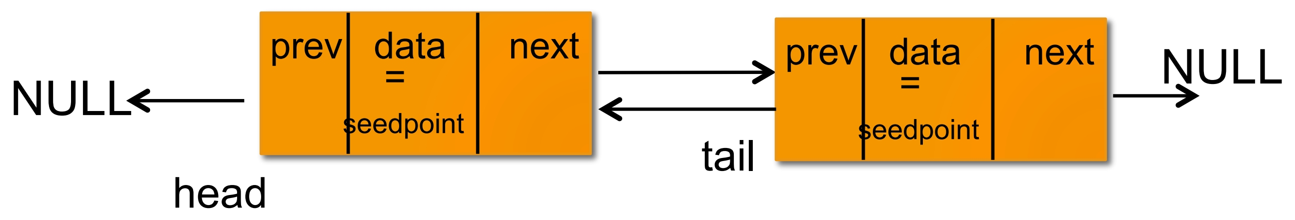
first call



seedpoint=seedpoint

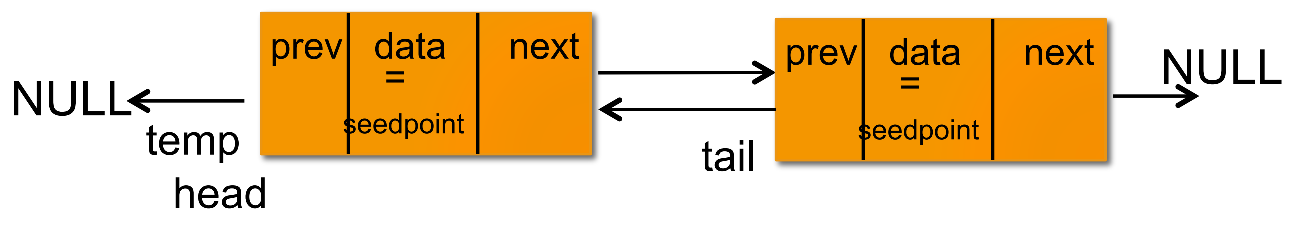
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
           seedpoint=temp->coord;
           (*head)=(*head)->next;
           free(temp);
            if((*head)!=NULL)
        →       (*head)->prev=NULL;
       return seedpoint;
   }
}
```

first call



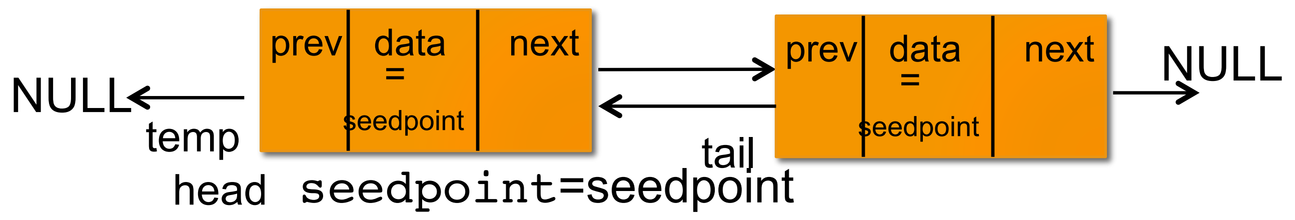
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
           seedpoint=temp->coord;
           (*head)=(*head)->next;
           free(temp);
            if((*head)!=NULL)
               (*head)->prev=NULL;
  →    return seedpoint;
  }
}
```

first call



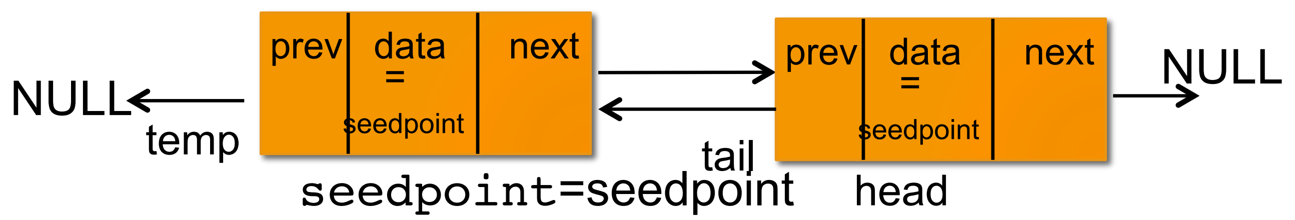
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{  struct coordinate seedpoint;
   struct double_list *temp;
   temp=*head;

   if(temp==NULL) //nothing on the list yet
   {
      printf("\n Nothing to exit..");
      exit(0);
   }
   else {
          seedpoint=temp->coord;
          (*head)=(*head)->next;
          free(temp);
           if((*head)!=NULL)
               (*head)->prev=NULL;
      return seedpoint;
  }
}
```

second call



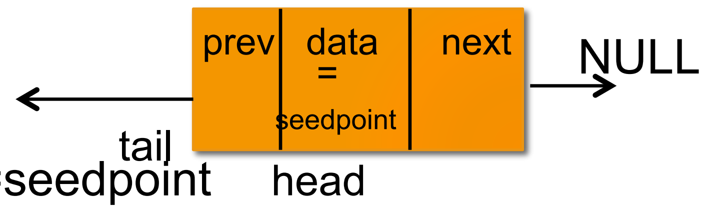
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
→           seedpoint=temp->coord;
            (*head)=(*head)->next;
            free(temp);
             if((*head)!=NULL)
                 (*head)->prev=NULL;
       return seedpoint;
   }
}
```

second call



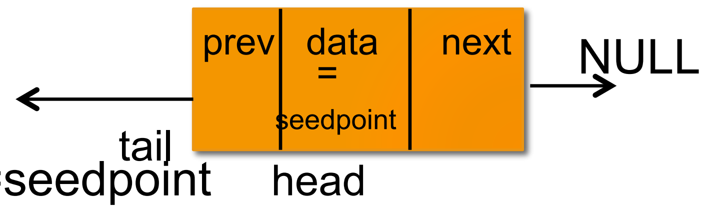
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
              seedpoint=temp->coord;
              (*head)=(*head)->next;
              free(temp);
               if((*head)!=NULL)
                   (*head)->prev=NULL;
       return seedpoint;
   }
}
```

second call



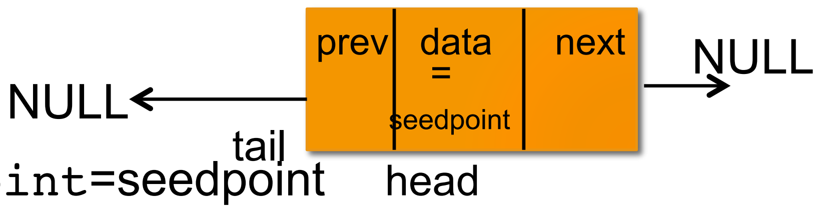
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{  struct coordinate seedpoint;
   struct double_list *temp;
   temp=*head;

   if(temp==NULL) //nothing on the list yet
   {
      printf("\n Nothing to exit..");
      exit(0);
   }
   else {
          seedpoint=temp->coord;
          (*head)=(*head)->next;
   →      free(temp);
           if((*head)!=NULL)
               (*head)->prev=NULL;
      return seedpoint;
  }
}
```

second call



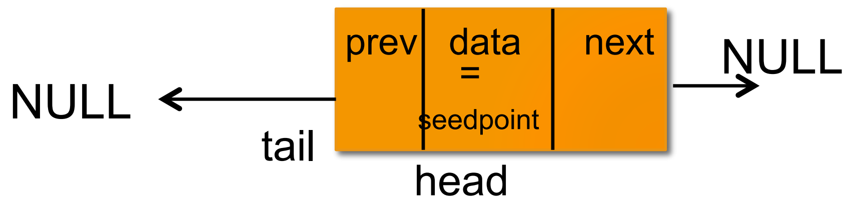
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{  struct coordinate seedpoint;
   struct double_list *temp;
   temp=*head;

   if(temp==NULL) //nothing on the list yet
   {
      printf("\n Nothing to exit..");
      exit(0);
   }
   else {
          seedpoint=temp->coord;
          (*head)=(*head)->next;
          free(temp);
    →      if((*head)!=NULL)
              (*head)->prev=NULL;
      return seedpoint;
  }
}
```

second call



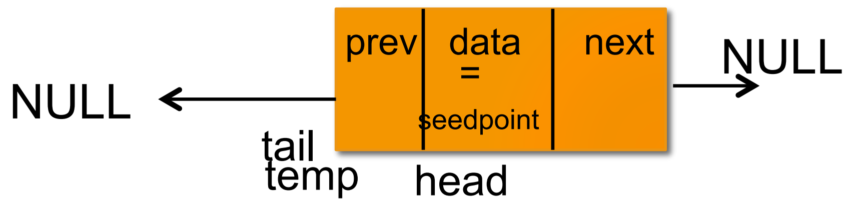
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
        printf("\n Nothing to exit..");
        exit(0);
    }
    else {
            seedpoint=temp->coord;
            (*head)=(*head)->next;
            free(temp);
             if((*head)!=NULL)
                 (*head)->prev=NULL;
        return seedpoint;
      }
}
```

second call



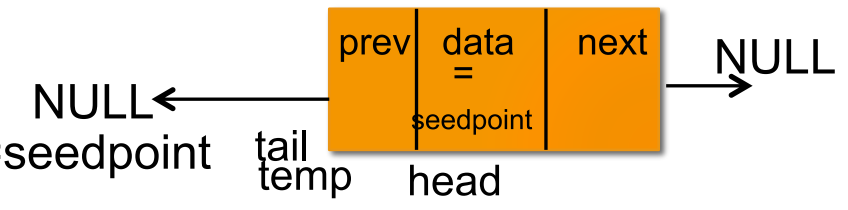
```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
           seedpoint=temp->coord;
           (*head)=(*head)->next;
           free(temp);
            if((*head)!=NULL)
                (*head)->prev=NULL;
  ---> return seedpoint;
   }
}
```

second call



```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
           seedpoint=temp->coord;
           (*head)=(*head)->next;
           free(temp);
            if((*head)!=NULL)
                (*head)->prev=NULL;
       return seedpoint;
   }
}
```

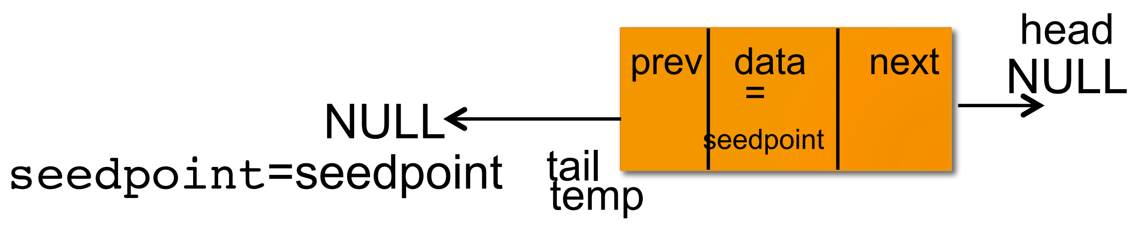
third call



seedpoint=seedpoint

```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
→             seedpoint=temp->coord;
              (*head)=(*head)->next;
              free(temp);
               if((*head)!=NULL)
                   (*head)->prev=NULL;
       return seedpoint;
   }
}
```

third call



```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
              seedpoint=temp->coord;
  →           (*head)=(*head)->next;
              free(temp);
               if((*head)!=NULL)
                  (*head)->prev=NULL;
       return seedpoint;
   }
}
```

third call

head
NULL

tail

seedpoint=seedpoint

```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
           seedpoint=temp->coord;
           (*head)=(*head)->next;
  →        free(temp);
            if((*head)!=NULL)
                (*head)->prev=NULL;
       return seedpoint;
   }
}
```

third call

head
NULL

tail

seedpoint=seedpoint

```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
       printf("\n Nothing to exit..");
       exit(0);
    }
    else {
           seedpoint=temp->coord;
           (*head)=(*head)->next;
           free(temp);
→           if((*head)!=NULL)
               (*head)->prev=NULL;
       return seedpoint;
   }
}
```

third call

head
NULL

tail

```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
        printf("\n Nothing to exit..");
        exit(0);
    }
    else {
            seedpoint=temp->coord;
            (*head)=(*head)->next;
            free(temp);
             if((*head)!=NULL)
                 (*head)->prev=NULL;
  $\longrightarrow$  return seedpoint;
   }
}
```

third call

head
NULL

tail

```
struct coordinate exit_queue(struct double_list **head) //call
by reference
{   struct coordinate seedpoint;
    struct double_list *temp;
    temp=*head;

    if(temp==NULL) //nothing on the list yet
    {
        printf("\n Nothing to exit..");
        exit(0);
    }
    else {
            seedpoint=temp->coord;
            (*head)=(*head)->next;
            free(temp);
             if((*head)!=NULL)
                 (*head)->prev=NULL;
        return seedpoint;
   }
}
```

what will happen on the fourth call?

DO NOT want program to access tail either

head
NULL

seedpoint=seedpoint

tail

```
struct coordinate exit_queue(struct double_list **head, struct
double_list **tail) //call by reference
{  struct coordinate seedpoint;
   struct double_list *temp;
   temp=*head;
   if(temp==NULL) //nothing on the list yet
   {
      printf("\n Nothing to exit..");
      exit(0);
   }
   else {
          seedpoint=temp->coord;
          (*head)=(*head)->next;
          free(temp);
           if((*head)!=NULL)
              (*head)->prev=NULL;
           else
               (*tail)=NULL;
      return seedpoint;
  }
}
```