	Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia	

Laboratorios de computación sala A y B

<i>Profesor:</i>	Ing. Adrian Ulises Mercado Martinez
<i>Asignatura:</i>	Estructura de datos y algoritmos I
<i>Grupo:</i>	13
<i>No de Práctica(s):</i>	12
<i>Integrante(s):</i>	Oscar Tovar Mendoza
<i>No. de Equipo de cómputo empleado:</i>	
<i>No. de Lista o Brigada:</i>	8
<i>Semestre:</i>	2
<i>Fecha de entrega:</i>	Domingo, 7 de junio de 2020

CALIFICACIÓN: _____

Introducción

La recursión es resolver un problema mediante la solución de un problema más pequeño del mismo, es muy usado en la programación y esta práctica tratara de este método, en el que se usará la forma con y sin recursividad.

Desarrollo

Como primera actividad creamos una lista pero con la función de borrar recursiva, usando el caso base cuando n->prev es igual a nulo

```
C ejercicio1.c > crear_nodo()
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  #include "ejercicio.h"
5
6  void insertar(INFO info, LIST *l){
7      if(l!=NULL){
8          if(l->head==NULL){
9              l->head = crear_nodo();
10             l->head->info = info;
11             return;
12         }
13         NODE *nuevo = crear_nodo();
14         nuevo->info = info;
15         nuevo->next = l->head;
16         l->head->prev = nuevo;
17         l->head = nuevo;
18     }
19 }
20
21 LIST *crear_lista(){
22     LIST *l = (LIST*) malloc(sizeof(LIST));
23     l->head = NULL;
24     l->tail = NULL;
25     return l;
26 }
27
28 void eliminar(LIST *l){
29     if(l->head!=NULL){
30         borrar_nodos(l->head);
31     }
32     free(l);
33 }
34
35 NODE crear_nodo(){
36     NODE *n = (NODE*) malloc(sizeof(NODE));
37     n->next = NULL;
38     n->prev = NULL;
39     strcpy(n->info.nombre, "");
40     strcpy(n->info.apellido, "");
41     return n;
42 }
43
44 void borrar_nodos(NODE *n){
45     if(n->next!=NULL){          //Caso recursivo
46         borrar_nodos(n->next);
47     }
48     n->prev = NULL;           //Caso base
49     free(n);
50 }
51
52 void imprimir(LIST *l){
53     for(NODE *i = l->head; i!=NULL; i = i->next){
54         printf("%s, %s\n", i->info.nombre, i->info.apellido);
55     }
56 }
```

```

C ejercicio.h > ...
1  #ifndef E1_H
2  #define E1_H
3
4  typedef struct _node NODE;
5
6  typedef struct _info{
7      char nombre[32];
8      char apellido[64];
9  } INFO;
10
11
12  struct _node{
13      INFO info;
14      NODE *next;
15      NODE *prev;
16  };
17
18  typedef struct _list{
19      NODE *tail;
20      NODE *head;
21  } LIST;
22
23  void insertar(INFO info, LIST *l);
24  LIST *crear_lista();
25  void eliminar(LIST *l);
26  void imprimir (LIST *l);
27
28  NODE crear_nodo();
29  void borrar_nodos(NODE *n);
30
31  #endif

```

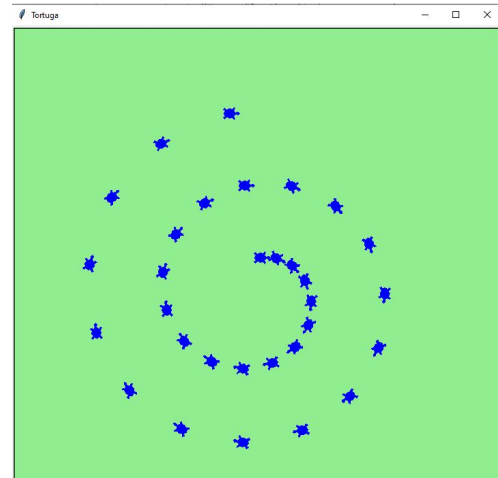
```

C main.c > main()
1  #include<stdio.h>
2  #include "ejercicio.h"
3  #include <string.h>
4
5  int main(){
6      LIST *lista;
7      INFO info;
8      strcpy(info.nombre, "nombre1");
9      strcpy(info.apellido, "apellido11 apellido12");
10
11      lista = crear_lista();
12      insertar(info, lista);
13      imprimir(lista);
14      eliminar(lista);
15      return 0;
16  }

```

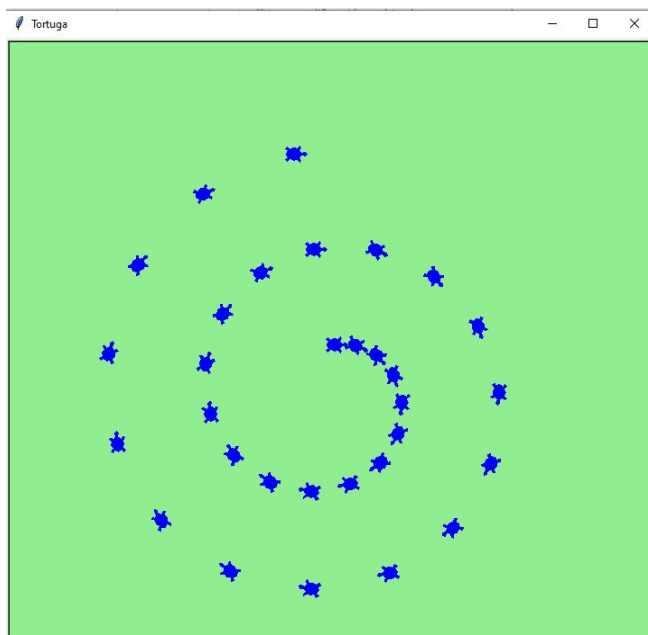
Como ejercicio 2 hicimos el recorrido de una tortuga sin el uso de la recursividad

```
ejercicio2.py > ...
1  import turtle
2
3  wn = turtle.Screen()
4  wn.bgcolor("lightgreen")
5  wn.title("Tortuga")
6  tess = turtle.Turtle()
7  tess.shape("turtle")
8  tess.color("blue")
9
10 tess.penup()
11 size = 20
12 for i in range(30):
13     tess.stamp()
14     size = size+3
15     tess.forward(size)
16     tess.right(24)
17
18 wn.mainloop()
```



Después hicimos el mismo ejercicio solo que esta vez con recursividad

```
ejercicio3.py > ...
1  import turtle
2  import argparse
3
4  def recorrido_recursivo(tortuga, espacio, huellas):
5      if huellas > 0:
6          tortuga.stamp()
7          espacio = espacio + 3
8          tortuga.forward(espacio)
9          tortuga.right(24)
10         recorrido_recursivo(tortuga, espacio, huellas-1)
11
12
13
14  '''
15  ap = argparse.ArgumentParser()
16  ap.add_argument("--huella", required = True, help="numero de huellas")
17  args = vars(ap.parse_args())
18  huellas = int(args["huella"])
19  '''
20
21  wn = turtle.Screen()
22  wn.bgcolor("lightgreen")
23  wn.title("Tortuga")
24  tess = turtle.Turtle()
25  tess.shape("turtle")
26  tess.color("blue")
27
28  tess.penup()
29  recorrido_recursivo(tess, 20,30)
30  wn.mainloop()
```



Conclusiones

- La recursividad es una herramienta muy útil y que ayuda a problemas complejos como lo son las torres de Hanoi y el problema de las 8 reinas, aunque es muy buena la recursividad, hay problemas que no necesariamente la necesitan mientras que en otros facilita el trabajo de alguna manera

(Oscar Tovar Mendoza)