	Carátula para entrega de prácticas	
Facultad de Ingeniería		Laboratorio de docencia

Laboratorio de computación Salas A y B

Profesor: Dr. Ismael Everardo Bárcenas Patiño

Asignatura: Estructuras de datos y algoritmos I

Grupo: 3

No. de práctica: 8

Integrantes:
10 Gurrión Aquino Carlos Ángel
14 León Ruiz Eduardo
16 Macías Niño Carmen Violeta
17 Marroquín García Ricardo
21 Montaño Torres Rodolfo Santiago

No. de equipo de cómputo empleado:

No. de brigada: 7

Semestre: 2022-1

Fecha de entrega: 26 de octubre de 2021

Observaciones:

Calificación:

Rúbrica de evaluación:

100	El programa cumple con todos los requerimientos
80-99	El programa cumple con la mayoría de los requerimientos
60-79	El programa cumple con algunos de los requerimientos
50-59	La lógica del programa es correcta, pero no corre o se cuelga

Código solución:

Enlace al archivo C guardado en nuestro repositorio de GitHub:
https://github.com/Rodolfo-Santiago/EDA1-2022-1-Practicas/blob/main/Pr%C3%A1ctica%208%20EDL:%20Lista%20doblemente%20ligada%20y%20doblemente%20ligada%20circular/eda1_p8_finalcode.c

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


#define   EN_JUEGO    1
#define   ELIMINADO   0
#define   MAXIMO      30
#define   MILISEG     300


struct Persona
{
      struct Persona* anterior_vivo;
      struct Persona* siguiente_vivo;

      int estado;
      char nombre[MAXIMO];

      struct Persona* anterior;
      struct Persona* siguiente;
};


int i;
int k;

int auxiliar_global;

struct Persona* inicio;
struct Persona* inicio_vivos;

struct Persona* temp_1;
struct Persona* temp_2;

int numero_personas;
int personas_en_juego;


void agregar_persona( );
void quitar_persona( );
void mostrar_todos_jugadores( );
void mostrar_jugadores_vivos( );
void ejecutar_juego( char* );
void resetear_juego( );

void descripcion_programa( );
void menu_acciones( );
void menu_acciones_dos( );

void limpiar_bufer( );
void barra_de_carga( );
```

```
void leer_entero( int* );
void leer_long( long long int* );



int main( )
{
      int opcion;

      inicio = NULL;
      inicio_vivos = NULL;

      numero_personas = 0;
      personas_en_juego = 0;
      auxiliar_global = 0;

      descripcion_programa( );

      while (1)
      {
             menu_acciones( );

             leer_entero( &opcion );

             switch (opcion)
             {
                    case 1:
                           agregar_persona( );
                           break;

                    case 2:
                           quitar_persona( );
                           break;

                    case 3:
                           if (inicio == NULL)
                           {
                                  printf("\n\nNo hay jugadores que mostrar.\n");
                                  break;
                           }

                           printf("\n\nEstos son los jugadores actualmente registrados:\n\n");

                           mostrar_jugadores_vivos( );
                           break;

                    case 4:
                           if (inicio == NULL)
                           {
                                  printf("\n\nNo se puede comenzar el juego sin tener jugadores
registrados.\n");
                                  break;
                           }

                           if (inicio == (*inicio).siguiente)
                           {
                                  printf("\n\nSolo hay una persona en el juego (%s).", (*inicio).nombre);
                                  printf("\nLe pedimos que agregue al menos otra persona para poder jugar.");
                                  break;
                           }

                           menu_acciones_dos( );
```

```
                        leer_entero( &opcion );

                        switch (opcion)
                        {
                                case 1:
                                        ejecutar_juego( "vivos y muertos" );
                                        break;

                                case 2:
                                        ejecutar_juego( "solamente vivos" );
                                        break;

                                default:
                                        printf("\nOpcion invalida. Vuelva a intentar.\n");
                                        break;
                        }
                        break;

                case 5:
                        printf("\nSeguro que desea salir:  0. Si   1. No\nOpcion: ");
                        if (leer_entero( &opcion ), opcion != 0) break;

                        printf("\nJuego terminado.");
                        return 0;

                default:
                        printf("\nOpcion invalida. Vuelva a intentar.\n");
                        break;
        }
        printf("\n");
    }

    return 0;
}



void agregar_persona( )
{
    int posicion_agregar;
    int var_auxiliar;


    temp_1 = (struct Persona*) calloc(1, sizeof(struct Persona));

    (*temp_1).estado = EN_JUEGO;

    printf("\n\nIngrese el nombre de la nueva persona: ");
    limpiar_bufer( );
    fgets((*temp_1).nombre, MAXIMO, stdin);
    (*temp_1).nombre[strlen((*temp_1).nombre) - 1] = '\0';


    if (inicio == NULL)
    {
        printf("\nAl ser la primera persona en el juego, %s sera colocada en la posicion 1.\n",
(*temp_1).nombre);

        inicio = temp_1;
        inicio_vivos = temp_1;

        (*temp_1).anterior = temp_1;
        (*temp_1).siguiente = temp_1;
```

```
		(*temp_1).anterior_vivo = temp_1;
		(*temp_1).siguiente_vivo = temp_1;

		++numero_personas;
		++personas_en_juego;

		return;
	}


	printf("\nIngrese la posicion (1-%d) donde desee colocar a %s: ", numero_personas + 1, (*temp_1).nombre);

	while (leer_entero( &var_auxiliar ), var_auxiliar < 1 || var_auxiliar > numero_personas + 1)
	{
		printf("No se puede colocar a la persona en esa posicion. Ingrese un valor valido.\n");
		printf("\nIngrese la posicion (1-%d) donde desee colocar a %s: ", numero_personas + 1,
(*temp_1).nombre);
	}

	if (var_auxiliar == 1) posicion_agregar = numero_personas + 1;
	else posicion_agregar = var_auxiliar;

	temp_2 = inicio;

	for (i = 1; i < posicion_agregar - 1; i++)
	{
		temp_2 = (*temp_2).siguiente;
	}

	(*temp_1).siguiente = (*temp_2).siguiente;
	(*(*temp_1).siguiente).anterior = temp_1;

	(*temp_2).siguiente = temp_1;
	(*temp_1).anterior = temp_2;

	(*temp_1).siguiente_vivo = (*temp_2).siguiente_vivo;
	(*(*temp_1).siguiente_vivo).anterior_vivo = temp_1;

	(*temp_2).siguiente_vivo = temp_1;
	(*temp_1).anterior_vivo = temp_2;


	if (var_auxiliar == 1) inicio = inicio_vivos = temp_1;

	++numero_personas;
	++personas_en_juego;
}



void quitar_persona( )
{
	int posicion_quitar;

	if (inicio == NULL)
	{
		printf("\n\nNo hay personas que eliminar.\n");
		return;
	}

	if (inicio == (*inicio).siguiente)
	{
```

```
            printf("\n\n%s era la unica persona en el juego, por lo que ha sido eliminada.",
(*inicio).nombre);

            temp_1 = inicio;

            inicio = NULL;
            inicio_vivos = NULL;

            free(temp_1);

            --numero_personas;
            --personas_en_juego;

            return;
      }


      printf("\n\nIngrese la posicion (1-%d) de la persona que desea sacar del juego: ", numero_personas);
      while (leer_entero( &posicion_quitar ), posicion_quitar < 1 || posicion_quitar > numero_personas)
      {
            printf("No se pueden saltar ese numero de personas en este juego. Vuelva a intentar.\n");
            printf("\nIngrese la posicion (1-%d) de la persona que desea sacar del juego: ", numero_personas);
      }

      if (posicion_quitar == 1) posicion_quitar = numero_personas + 1;

      temp_1 = inicio;
      for (i = 1; i < posicion_quitar - 1; i++)
      {
            temp_1 = (*temp_1).siguiente;
      }

      if ((*temp_1).siguiente == inicio) inicio = inicio_vivos = (*inicio).siguiente;

      temp_2 = (*temp_1).siguiente;

      (*temp_1).siguiente = (*temp_2).siguiente;
      (*(*temp_1).siguiente).anterior = temp_1;

      (*temp_1).siguiente_vivo = (*temp_2).siguiente_vivo;
      (*(*temp_1).siguiente_vivo).anterior_vivo = temp_1;

      printf("\n%s ha sido sacado del juego.", (*temp_2).nombre);

      free(temp_2);

      --numero_personas;
      --personas_en_juego;
}



void mostrar_todos_jugadores( )
{
      temp_2 = inicio;

      printf("\t\tPOSICION\t\tNOMBRE\t\t ESTADO\n");

      for (k = 1; k <= numero_personas; k++)
      {
            if ((*temp_2).estado == ELIMINADO)
            {
                  printf("\t%12d %25s      [Eliminado] X\n", k, (*temp_2).nombre);
```

```
		}

		else if ((*temp_2).estado == EN_JUEGO && auxiliar_global == 1)
		{
			printf("\t%12d %25s        [Ganador] !\n", k, (*temp_2).nombre);
		}

		else if ((*temp_2).estado == EN_JUEGO)
		{
			printf("\t%12d %25s       [En juego] +\n", k, (*temp_2).nombre);
		}

		temp_2 = (*temp_2).siguiente;
	}
}



void mostrar_jugadores_vivos( )
{
	temp_2 = inicio_vivos;

	printf("\t\tPOSICION\t\tNOMBRE\t\t ESTADO\n");

	for (k = 1; k <= personas_en_juego; k++)
	{
		if ((*temp_2).estado == EN_JUEGO && auxiliar_global == 1)
		{
			printf("\t%12d %25s        [Ganador] !\n", k, (*temp_2).nombre);
		}

		else if ((*temp_2).estado == EN_JUEGO)
		{
			printf("\t%12d %25s       [En juego] +\n", k, (*temp_2).nombre);
		}

		temp_2 = (*temp_2).siguiente_vivo;
	}
}



void ejecutar_juego( char mostrar[15] )
{
	int posicion_inicio;
	long long int numero_saltos;
	int var_auxiliar;

	printf("\n\nEstos son los jugadores inscritos al juego:\n");
	mostrar_jugadores_vivos( );


	printf("\nIngrese la posicion (1-%d) donde empezaremos el juego: ", numero_personas);
	while (leer_entero( &posicion_inicio ), posicion_inicio < 1 || posicion_inicio > numero_personas)
	{
		printf("La posicion que ingreso no existe. Vuelva a intentar.\n");
		printf("\nIngrese la posicion (1-%d) donde empezaremos el juego: ", numero_personas);
	}


	printf("\nIngrese el numero de personas que \"saltaremos\" en cada eliminacion: ");
	while (leer_long( &numero_saltos ), numero_saltos == 0)
	{
```

```
        printf("No se pueden saltar ese numero de personas en este juego. Vuelva a intentar.\n");
        printf("\nIngrese el numero de personas que \"saltaremos\" en cada eliminacion: ");
}


temp_1 = inicio;
for (i = 1; i < posicion_inicio; ++i)
{
        temp_1 = (*temp_1).siguiente;
}


printf("\n\nEstado inicial del juego (empezaremos a contar desde %s)\n", (*temp_1).nombre);
mostrar_jugadores_vivos( );


for (i = 1; i < numero_personas; ++i)
{
        if (personas_en_juego == numero_personas) var_auxiliar = 0;
        else var_auxiliar = 1;

        if (numero_saltos > 0)
        {
                for (k = 1; k <= numero_saltos - var_auxiliar; ++k)
                {
                        temp_1 = (*temp_1).siguiente_vivo;
                }
        }
        else if (numero_saltos < 0)
        {
                for (k = -1; k >= numero_saltos + var_auxiliar; --k)
                {
                        temp_1 = (*temp_1).anterior_vivo;
                }
        }


        --personas_en_juego;


        //Eliminar
        if (temp_1 == inicio_vivos)  inicio_vivos = (*inicio_vivos).siguiente_vivo;


        (*(*temp_1).anterior_vivo).siguiente_vivo = (*temp_1).siguiente_vivo;

        (*(*temp_1).siguiente_vivo).anterior_vivo = (*temp_1).anterior_vivo;

        (*temp_1).estado = ELIMINADO;


        if (numero_saltos > 0) temp_1 = (*temp_1).siguiente_vivo;
        else if (numero_saltos < 0) temp_1 = (*temp_1).anterior_vivo;



        printf("\nEliminacion #%d", i);
        barra_de_carga( );

        if (i == numero_personas - 1) auxiliar_global = 1;

        if (mostrar == "vivos y muertos") mostrar_todos_jugadores( );
        else if (mostrar == "solamente vivos") mostrar_jugadores_vivos( );
```

```
	}

	printf("\n\t\t%s fue el ultimo sobreviviente y ha ganado el juego.",
(*(*inicio_vivos).siguiente_vivo).nombre);
	printf("\n\n\nEste juego ha terminado. Puede comenzar un nuevo juego con el siguiente menu:");

	resetear_juego( );
}



void resetear_juego( )
{
	temp_1 = inicio;
	temp_2 = inicio;

	inicio = NULL;
	inicio_vivos = NULL;

	for (i = 1; i < numero_personas; i++)
	{
		temp_1 = (*temp_1).siguiente;

		free(temp_2);

		temp_2 = temp_1;
	}

	free(temp_2);

	numero_personas = 0;
	personas_en_juego = 0;

	auxiliar_global = 0;
}



void barra_de_carga( )
{
	int j;

	int numero_rayas;
	int espacio_disponible = 50;

	int milisegundos_limite;
	int milisegundos;

	printf("\nCargando        ");

	numero_rayas = rand() % espacio_disponible;
	espacio_disponible -= numero_rayas;
	milisegundos_limite = rand() % MILISEG;
	milisegundos = rand() % milisegundos_limite;

	for (j = 0; j < numero_rayas; j++)
	{
	  Sleep(milisegundos);
		printf("%c", 220);
	}

	if (espacio_disponible == 0) return;
```

```
numero_rayas = rand() % espacio_disponible;
espacio_disponible -= numero_rayas;
milisegundos_limite = rand() % MILISEG;
milisegundos = rand() % milisegundos_limite;

for (j = 0; j < numero_rayas; j++)
{
  Sleep(milisegundos);
       printf("%c", 220);
}

if (espacio_disponible == 0) return;

numero_rayas = rand() % espacio_disponible;
espacio_disponible -= numero_rayas;
milisegundos_limite = rand() % MILISEG;
milisegundos = rand() % milisegundos_limite;

for (j = 0; j < numero_rayas; j++)
{
  Sleep(milisegundos);
       printf("%c", 220);
}

if (espacio_disponible == 0) return;

numero_rayas = rand() % espacio_disponible;
espacio_disponible -= numero_rayas;
milisegundos_limite = rand() % MILISEG;
milisegundos = rand() % milisegundos_limite;

for (j = 0; j < numero_rayas; j++)
{
  Sleep(milisegundos);
       printf("%c", 220);
}

if (espacio_disponible == 0) return;

numero_rayas = rand() % espacio_disponible;
espacio_disponible -= numero_rayas;
milisegundos_limite = rand() % MILISEG;
milisegundos = rand() % milisegundos_limite;

for (j = 0; j < numero_rayas; j++)
{
  Sleep(milisegundos);
       printf("%c", 220);
}

if (espacio_disponible == 0) return;

numero_rayas = rand() % espacio_disponible;
espacio_disponible -= numero_rayas;
milisegundos_limite = rand() % MILISEG;
milisegundos = rand() % milisegundos_limite;

for (j = 0; j < numero_rayas; j++)
{
  Sleep(milisegundos);
       printf("%c", 220);
}
```

```
	printf("\n");
}



void leer_entero( int* entero )
{
	if (scanf("%d", entero) != 1)
	{
		printf("\nEl tipo de dato que ingreso no es un numero entero. Reinicie el juego.");
		exit(-1);
	}
}



void leer_long( long long int* entero )
{
	if (scanf("%lld", entero) != 1)
	{
		printf("\nEl tipo de dato que ingreso no es un numero entero. Reinicie el juego.");
		exit(-1);
	}
}


void limpiar_bufer( )
{
	while (fgetc(stdin) != '\n')
		;
}



void descripcion_programa( )
{
	printf("BIENVENIDO AL JUEGO \"EL ULTIMO SOBREVIVIENTE\"\n");
	printf("\n\tDado un conjunto de personas ordenadas de forma circular, el juego consiste en, ");
	printf("comenzando en una cierta posicion del circulo, eliminar personas cada cierto periodo");
	printf(" hasta que solo quede una persona en juego: la sobrevieviente. Tanto la cantidad de ");
	printf("personas en el juego, como sus nombres, la posicion de inicio y el periodo son ");
	printf("proporcionados por el usuario. El menu de acciones disponibles es el siguiente.");
}



void menu_acciones( )
{
	printf("\n\nMenu de acciones:\n\t1. Agregar a una nueva persona al juego.\n\t2. Sacar a una ");
	printf("persona del juego.\n\t3. Mostrar a las personas que ya estan formadas.\n\t4. Comenza");
	printf("r el juego.\n\t5. Salir del juego.\n\n\tElija la opcion que desea ejecutar: ");
}



void menu_acciones_dos( )
{
	printf("\n\nAntes de comenzar el juego, seleccione la forma en que quiere visualizar a los");
	printf(" jugadores despues de cada etapa del juego:\n\t1. Ver personas eliminadas y en ");
	printf("juego.\n\t2. Ver solo personas en juego.\n\n\tElige la opcion que desea ejecutar: ");
}
```