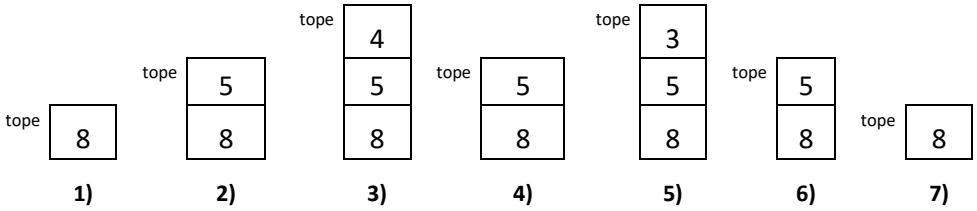
SOLEMNE1:

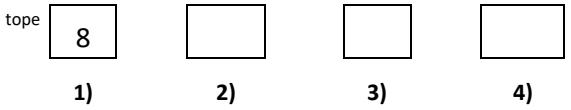
EJERCICIO 1: 2P

7) ***Pop(Pop(PUSH(3,Pop(PUSH(4,PUSH(5,PUSH(8,P)))))))***
6) ***Pop(PUSH(3,Pop(PUSH(4,PUSH(5,PUSH(8,P))))))***
5) ***PUSH(3,Pop(PUSH(4,PUSH(5,PUSH(8,P)))))***
4) ***Pop(PUSH(4,PUSH(5,PUSH(8,P))))***
3) ***PUSH(4,PUSH(5,PUSH(8,P)))***
2) ***PUSH(5,PUSH(8,P))***
1) ***PUSH(8,P)***



0.7P

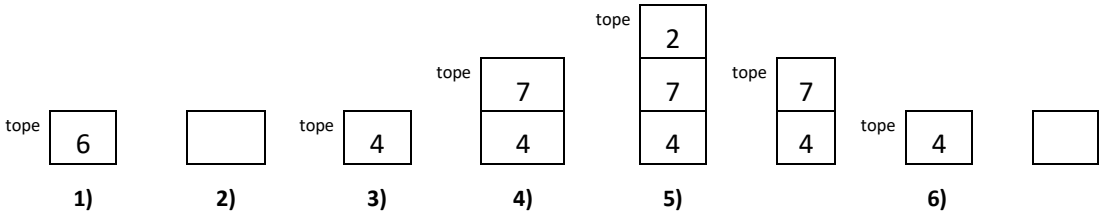
4) ***EsVacia(Pop(Pop(PUSH(8,P))))***
3) ***Pop(Pop(PUSH(8,P)))***
2) ***Pop(PUSH(8,P))***
1) ***PUSH(8,P)***



"Es vacia" True

0.4P

6) ***listarPila(PUSH(2,PUSH(7,PUSH(4,Pop(PUSH(6,P))))))***
5) ***PUSH(2,PUSH(7,PUSH(4,Pop(PUSH(6,P)))))***
4) ***PUSH(7,PUSH(4,Pop(PUSH(6,P))))***
3) ***PUSH(4,Pop(PUSH(6,P)))***
2) ***Pop(PUSH(6,P))***
1) ***PUSH(6,P)***



0.9P

Al listar se desempila:

EJERCICIO 2: 3P

PROTOTIPOS: 0.4P

```
Cola *crearPila();
Info *crearInfo(int pval);
void push(Pila *p, int pval);
Info pop(Pila *p);
void recorrerPila(Pila *p);
void destruirPila(Pila *p);
Pila *poner_nElems_AlFinal(Pila *pila, int n)
```

Declaración var y crear auxiliares	0.3P
ciclo apilar AUX1 con n elementos de P	0.5P
ciclo apilar AUX2 con tam elem de P	0.5P
Apilar AUX1 y luego Apilar AUX2	0.4P

ESTRUCTURAS: 0.5P

```
typedef struct info {
    int valor;
} Info;
typedef struct nodo {
    Info *elem;
    struct nodo *sgte;
} Nodo;
typedef struct pila {
    Nodo *tope;
    int tam;
} Pila;
```

MAIN: 0.4P

```
int main(int argc, char *argv[]){
    Pila *P;
    P = crearPila();
    push(P,0);
    push(P,2);
    push(P,4);
    push(P,3);
    push(P,5);
    push(P,6);
    int n = 2;
    P = poner_nElems_AlFinal(P,n);
    recorrerPila(P);
    destruirPila(P);
}
```

FUNCIÓN SOLICITADA: 1.7P

```
Pila *poner_nElems_AlFinal(Pila *pila, int n){
    Pila *pAux1=crearPila();
    Pila *pAux2=crearPila();
    int i=0;
    //sacar n elementos y ponerlos en pAUX1
    for(i=1;i<=n;i++){
      push(pAux1, pop(pila)->valor); //el pop retorna un info
    }
    //sacar los elementos restantes de p y ponerlos en pAUX2
    int tam = pila->tam;//pila-tam se modifica por eso se ocupa tam
    for(i=1;i<=tam;i++){
      push(pAux2, pop(pila)->valor);
    }
    //apilar p con elementos de pAUX1
    int tamAux1 = pAux1->tam;
    for(i=1;i<=tamAux1;i++){
      push(pila, pop(pAux1)->valor);
    }
    //apilar p con elementos de pAUX2
    int tamAux2 = pAux2->tam;
    for(i=1;i<=tamAux2;i++){
      push(pila, pop(pAux2)->valor);
    }
    return pila;
}
```

EJERCICIO 3: 3P

PROTOTIPOS: 0.4P

```
Cola *crearCola();
Info *crearInfo(int pval);
void encolar(Cola *pc, int pval);
int desencolar(Cola *pc);
void recorrerCola(Cola *pc);
void destruirCola(Cola *pc);
void promediarPorCada_m(Cola *pc, int m);
```

Definir e inicializar variables y cuenta elementos procesados	0.5
Ciclo acumula valor y desencola para cada grupo m ..	0.5
Ciclo calcula promedio y encola en la misma cola	0.5
Controla el tamaño de cada grupo	0.5

ESTRUCTURAS: 0.2P

```
typedef struct info {
    int valor;
} Info;
typedef struct nodo {
    Info *elem;
    struct nodo *sgte;
} Nodo;
typedef struct cola {
    Nodo *frente;
    Nodo *fin;
    int tam;
} Cola;
```

MAIN: 0.4P

```
int main(int argc, char *argv[]){
    Cola *c;
    c = crearCola();
    encolar(c,1);
    encolar(c,3);
    encolar(c,2);
    encolar(c,4);
    encolar(c,5);
    encolar(c,2);
    encolar(c,2);
    encolar(c,3);
    encolar(c,5);
    encolar(c,7);
    encolar(c,4);
    encolar(c,3);
    encolar(c,2);
    encolar(c,2);
    int m=3;
    promediarPorCada_m(c,m);
    recorrerCola(c);
    destruirCola(c);
    return 0;
}
```

FUNCIÓN SOLICITADA: 2.0P

```
void promediarPorCada_m(Cola *pc, int m){
    int i;
    int tam = pc->tam;
    while (tam!=0) {//o tambien <0
        int sum=0; int cuenta = 0;
        for (i=1;i<=m;i++){//se saca el promedio de los m elementos
            sum = sum + pc->frente->elem->valor;
            desencolar(pc);
            cuenta++; //el último grupo puede ser distinto de m
        }
        double val = (double)sum/(double)cuenta;
        encolar(pc,round(val));
        //
        tam = tam - cuenta; //descuento los desencolados
        //
        if (tam<m&&tam>0)// porque en el último grupo pueden quedar
            m=tam; // menos elementos que m
    }
}
```

EJERCICIO 4: 2P

PROTOTIPOS: 0.2P

```
Lista *crearLista();
Info *crearInfo(int pval);
void insertarElemento(Lista *lis, int pval);
void recorrerLista(Lista *lis);
void destruirLista(Lista *lis);
Lista *multiplicarListas(Lista *l1, Lista *l2);
```

Retornar L1 ó L2 o NULL o L3 según corresponda	0.2
Declara auxs, crear lista res e inicializar auxs	0.3
Recorrer lista 1 (mover aux1)	0.2
Para cada nodo de lista 1 recorrer aux2 (mover)	0.2
Multiplicar dentro del ciclo lista 2	0.2
Volver al inicio de lista 2 por cada recorrido de l1	0.2

ESTRUCTURAS:0.2P

```
typedef struct info {
    int valor;
} Info;
typedef struct nodo {
    Info *elem;
    struct nodo *sgte;
} Nodo;
typedef struct lista {
    Nodo *ini;
    Nodo *fin;
    int tam;
} Lista;
```

MAIN: 0.3P

```
int main(int argc, char *argv[]){
    Lista *L1;
    L1 = crearLista();
    Lista *L2;
    L2 = crearLista();
    insertarElemento(L1,1);
    insertarElemento(L1,3);
    insertarElemento(L1,5);
    insertarElemento(L1,0);
    insertarElemento(L2,4);
    insertarAlInicio(L2,3);
    insertarAlInicio(L2,2);
    insertarAlInicio(L2,1);
    Lista *L3;
    L3 = crearLista();
    L3=multiplicarListas(L1,L2);
    recorrerLista(L3);
    destruirLista(L1);
    destruirLista(L2);
    destruirLista(L3);
}
```

FUNCIÓN SOLICITADA: 1.3P

```
Lista *multiplicarListas(Lista *l1, Lista *l2){
    if ((l1->tam==0)&& (l2->tam==0)) {
        printf("Listas 1 y 2 vacias");
        return NULL;
    }else{
      if ((l1->tam==0)&& (l2->tam!=0)) {
        printf("Listas 1 vacia");
         return l2;
      }else{
        if ((l1->tam!=0)&& (l2->tam==0)) {
            printf("Listas 2 vacia");
            return l1;
        }else{
            Nodo *aux1;
            Nodo *aux2;
            Lista *l3= crearLista();
            aux1 = l1->ini;
           while (aux1 != NULL){
              aux2 = l2->ini;
              while (aux2 != NULL){
                 insertarElemento(l3,aux1->elem->valor*aux2->elem->valor);
                 aux2 = aux2->sgte;
              }
              aux1 = aux1->sgte;
           }
          return l3;
        }
      }
    }
}
```

ESCALA DE NOTAS:

Puntaje	Nota
0.0	1.5
1.0	1.5
2.0	1.6
3.0	1.6
4.0	1.7
5.0	1.7
6.0	1.8
7.0	1.8
8.0	1.8
9.0	1.9

Puntaje	Nota
10.0	1.9
11.0	2.0
12.0	2.0
13.0	2.0
14.0	2.1
15.0	2.1
16.0	2.2
17.0	2.2
18.0	2.3
19.0	2.3

Puntaje	Nota
20.0	2.3
21.0	2.4
22.0	2.4
23.0	2.5
24.0	2.5
25.0	2.5
26.0	2.6
27.0	2.6
28.0	2.7
29.0	2.7

Puntaje	Nota
30.0	2.8
31.0	2.8
32.0	2.8
33.0	2.9
34.0	2.9
35.0	3.0
36.0	3.0
37.0	3.0
38.0	3.1
39.0	3.1

Puntaje	Nota
40.0	3.2
41.0	3.2
42.0	3.3
43.0	3.3
44.0	3.3
45.0	3.4
46.0	3.4
47.0	3.5
48.0	3.5
49.0	3.5

Puntaje	Nota
50.0	3.6
51.0	3.6
52.0	3.7
53.0	3.7
54.0	3.8
55.0	3.8
56.0	3.8
57.0	3.9
58.0	3.9
59.0	4.0

Puntaje	Nota
60.0	4.0
61.0	4.1
62.0	4.2
63.0	4.2
64.0	4.3
65.0	4.4
66.0	4.5
67.0	4.5
68.0	4.6
69.0	4.7

Puntaje	Nota
70.0	4.8
71.0	4.8
72.0	4.9
73.0	5.0
74.0	5.1
75.0	5.1
76.0	5.2
77.0	5.3
78.0	5.4
79.0	5.4

Puntaje	Nota
80.0	5.5
81.0	5.6
82.0	5.7
83.0	5.7
84.0	5.8
85.0	5.9
86.0	6.0
87.0	6.0
88.0	6.1
89.0	6.2

Puntaje	Nota
90.0	6.3
91.0	6.3
92.0	6.4
93.0	6.5
94.0	6.6
95.0	6.6
96.0	6.7
97.0	6.8
98.0	6.9
99.0	6.9

Puntaje	Nota
100.0	7.0

ENUNCIADOS SOLEMNE 1:

Ejercicio 1:

Indique el resultado de las operaciones que indican en las expresiones siguientes. Evalúe paso a paso (con dibujos claros y explicativos) cada expresión hasta llegar al resultado:

a) pop(pop(push(3,pop(push(4,push(5,push(8,p)))))))
b) esVacia(pop(pop(push(8,p))))
c) listarPila(push(2,push(7,push(4,pop(push(6,p))))))

Para cada ejercicio que se presenta a continuación (ejercicio del 2 al 4), usted debe realizar lo siguiente, en el mismo orden:

a) Defina los typedef utilizados (se exige un typedef Info para la información del nodo).
b) Indique los prototipos de las funciones elementales que ya están implementadas y de la nueva función solicitada.
c) Implementar el main. Éste debe llamar a todas las funciones necesarias previamente, antes de ejecutar la función solicitada.
d) Implementar la función solicitada.
e) Implementar las funciones elementales requeridas que NO están consideradas en el enunciado.

Ejercicio 2:
Escriba una función que reciba una Pila y un número entero n. Esta función debe poner los primeros n elementos de la Pila al final de la misma y retornar esta misma pila con los elementos modificados. Por ejemplo, si la Pila P es P = {6, 5, 3, 4, 2, 0} y n es 2, debe retornar P= {3, 4, 2, 0, 6, 5}.
Suponga que ya están implementadas las funciones crear pila, push y pop.

Ejercicio 3:
Escriba una función que reciba una Cola y un número entero m. Esta función debe calcular el promedio (redondear a entero) por cada m elementos y dejar el resultado en la misma cola respetando la ubicación. Además, debe considerar que el último grupo podría no contener m elementos, en este caso, el promedio debe considerar los elementos que logre agrupar. Por Ejemplo:
Si la Colas es C= {1,3,2,4,5,2,2,3,5,7,4,3,2,2} y si m=3, la cola resultante sería: C = {2, 4, 3, 5, 2}.
Suponga que ya están implementadas las funciones crear cola, encolar y desencolar.

Ejercicio 4:
Escribir una función que reciba dos listas L1 y L2 y retorne una tercera lista L, donde L1={a1,a2,..,aN} y L2={b1,b2,...,bM} y L es el resultado de multiplicar L1 x L2 de la siguiente forma: L = {a1b1, a1b2,...,a1bM, a2b1,a2b2,...,a2bM, ... aN b1,aN b2,...,aN bM}.
Por ejemplo, si L1={1, 3, 5, 0} y L2={1, 2, 3, 4} entonces L sería: L = {1, 2, 3, 4, 3, 6, 9, 12, 5, 10, 15, 20, 0, 0, 0, 0}.
Suponga que ya están implementadas las funciones crear lista, insertar y eliminar.