CC3301 Programación de Software de Sistemas

Control 1 – Semestre Primavera 2012

Prof.: Luis Mateu

Pregunta 1

Parte a.- Programe la función *uint2hexa* que convierte un entero de 32 bits sin signo a un *string* con su representación en hexadecimal. La función recibe como parámetros el entero y un arreglo de caracteres con suficiente espacio. Ud. **no puede usar los operadores /, % y ***, pero sí puede recurrir a los operadores >>, <<, &, | y ~. El encabezado de la función es:

```
void uint2hexa(unsigned int x, char h[ ]);
```

Ejemplo de uso:

```
char hexa[9];
uint2hexa(2049796943, hexa);
printf("%s\n", hexa); /* debe mostrar 7a2d6b4f */
```

Parte b.- Programe la función *bit_mas_significativo* que entrega la posición del bit más significativo de un entero de 32 bits sin signo. Ud. **debe ser eficiente usando una búsqueda binaria para determinar el resultado en un ciclo de no más de 5 iteraciones**. Ud. no puede realizar solo desplazamientos de a un bit para determinar el resultado. El encabezado de la función es:

```
int bit_mas_sigfinificativo(unsigned int x);
```

Ejemplos de uso:

Invocación	*Resultado*
`bit_mas_significativo(0)`	-1
`bit_mas_significativo(1)`	0
`bit_mas_significativo(0x10A01)`	16
`bit_mas_significativo(0x1F2)`	8
`bit_mas_significativo(0x2A31)`	13
`bit_mas_significativo(0xF1001)`	19
`bit_mas_significativo(0xFFFFFFFF)`	31

Pregunta 2

Se define de la siguiente forma la estructura de una lista simplemente enlazada:

```
typedef struct nodo {
  int info;
  struct node *prox;
} Nodo;
```

Parte i.- Programe la función *duplicar* que recibe un puntero al primer nodo de una lista y entrega una copia de esa lista con la misma información. La nueva lista no puede compartir ningún nodo con la lista original. Además Ud. ***debe pedir el espacio en memoria para la nueva lista con un solo malloc***, de tal forma que un solo free permita liberar el espacio de la lista completa. El encabezado de la función es:

```
Nodo *duplicar(Nodo *cabeza);
```

Hint: Determine primero el número de nodos de la lista original y luego invoque malloc pidiendo memoria para todos los nodos de la nueva lista.

Parte ii.- Programe la función *contiene_ciclo* que determina si una lista enlazada contiene un ciclo. Es decir si algún nodo apunta a un nodo previo en la lista (no necesariamente la cabeza). Ud. **no puede usar *malloc* ni tampoco modificar la lista** (no puede agregar campos, alterar *prox* o *info*). El encabezado de la función es:

```
int contiene_ciclo(Nodo *cabeza);
```

Hint: Programe una función auxiliar recursiva que reciba (1) un puntero a un nodo de la lista enlazada, y (2) un puntero a un nodo de una segunda lista enlazada en la que almacena los punteros de los nodos ya visitados de (1). Como no puede usar malloc, declare un nodo de la segunda lista enlazada como variable local (automática) de la función recursiva. Ejemplo de uso:

```
Nodo nodos[ ]= { {10, &nodos[1]}, {5, &nodos[2]},
                 { 30, NULL} };
if (contiene_ciclo(&nodos[0])) /* falso */
  ...
nodos[2].prox= &nodos[1]; /* arma el ciclo */
if (contiene_ciclo(&nodos[0])) /* verdadero */
  ...
```