Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC3301 - Programación de Software de Sistemas

Auxiliar 11

Preparación para el Examen

Profesor: Luis Mateu

Auxiliares: Gerard Cathalifaud
Vicente González
Joaquín López
Rodrigo Urrea

Semestre: Primavera 2023

Preguntas

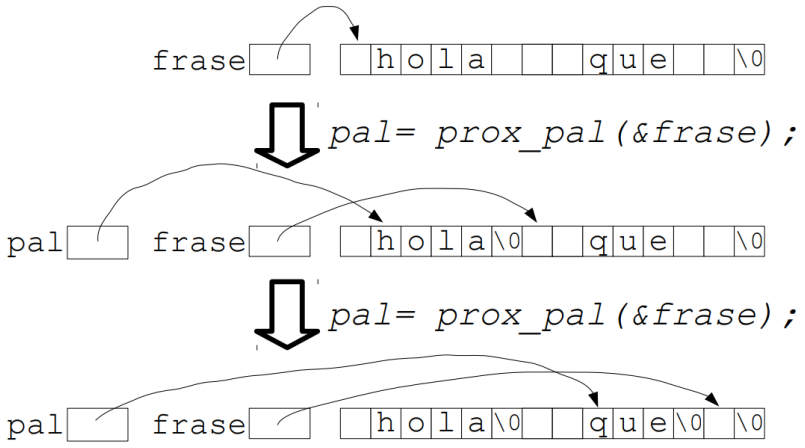
P1. (Examen 2016-P) Manejo de strings

Programe la siguiente función:

```
char *prox_pal(char **pfrase);
```

El parámetro `*pfrase` apunta a un string de C que contiene múltiples palabras separadas por uno o más espacios en blancos. Esta función debe retornar la primera palabra de la frase y entregar en `*pfrase` lo que quedo ´de la frase.

La siguiente figura muestra 2 ejemplos de uso. Los punteros `pal` y `frase` son de tipo `char*`. Inicialmente `frase` apunta al string `" hola que "` y termina apuntando a `" "`.



La primera llamada retorna `"hola"` y la segunda `"que"`. Observe que Ud. debe modificar el string recibido colocando la terminación del string (`'\0'`) para la palabra retornada. Si no quedan palabras en `frase`, se debe retornar `NULL`.

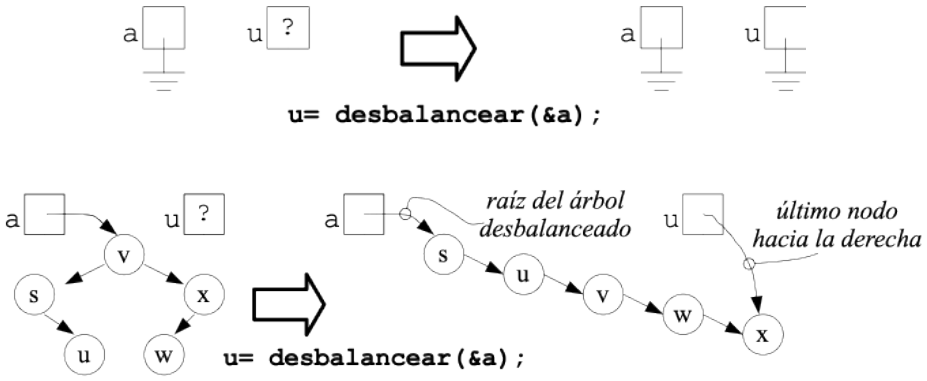
Restricciones: No use el operador de subindicación de arreglos `[ ]` ni su equivalente `*(p+i)`, use aritmética de punteros. No use `malloc`.

P2. (CR 2023-O) Estructuras

Programe la función:

```
Nodo *desbalancear(Nodo **pa);
```

La estructura `Nodo` es la usual con campos `izq` y `der` para los subárboles izquierdo y derecho. La función `desbalancear` recibe en `*pa` un árbol binario de búsqueda (ABB) y entrega en el mismo `*pa` un árbol equivalente pero desbalanceado al extremo a la derecha, es decir un árbol en el que todos los subárboles izquierdos son `NULL`. Además retorna la dirección del nodo de más a la derecha del árbol resultante. La función debe reutilizar los mismos nodos del árbol original. No se puede pedir memoria con `malloc`. La figura muestra 2 ejemplos de uso. Las variables `a` y `u` son de tipo `Nodo *`.



Metodología obligatoria: Sea `a=*pa`. Considere el caso en que a no es `NULL`, el subárbol izquierdo de `a` no es `NULL` y el derecho sí es `NULL`. Desbalancee recursivamente el subárbol izquierdo de `a` y obtendrá `izq`, el subárbol izquierdo desbalanceado, y `ultIzq`, el último nodo de ese subárbol. Enlace el subárbol derecho de `ultIzq` con `a`, los subárboles izquierdo y derecho de `a` con `NULL`. El nuevo valor de `*pa` debe ser `izq` y retorne `a`. Proceda de manera similar con los otros casos.

P3. (C2 2022-P Arquitectura) Assembler

Traduzca la función `g` de más abajo a assembler Risc-V. Optimice el código en assembler para reducir la cantidad de instrucciones.

```
typedef struct nodo {
  int x, y;
  stuct nodo *izq, *der;
} Nodo;

Nodo *g(Nodo *a, int z, int *pres) {
  while (a != NULL && a->x != z) {
    if (z < a->x)
      a = a->izq;
```

```
    else
      a = a->der;
  }

  if (a != NULL) {
    *pres = a->y;
  }
  return a;
}
```

P4. Fork y señales

(a) (C3 2018-P) Programe la función:

```
char *ultimaDireccionValida(char *ptr);
```

Esta función debe entregar la última dirección válida que se puede leer a partir de `ptr`. Para calcularla lea el carácter apuntado por `ptr` e incremente `ptr` en 1. Repita esta lectura e incremento indefinidamente hasta que se produzca el *segmentation fault*. Capture la señal `SIGSEGV`. Resguarde `ptr` en una variable global. El último valor de `ptr` menos 1 es la última dirección válida.

(b) (C3 2014-O) Ud. dispone de la función integral del control 2:

```
typedef double (*Funcion)(void *ptr, double x);

double integral(Funcion f, void *ptr, double xi, double xf, int n);
```

Recuerde que `integral(f, ptr, xi, xf, n)` calcula $\int_{x_f}^{x_i} f(\texttt{ptr}, x)dx$ numéricamente.

Programe la función:

```
double integral_par(Funcion f, void *ptr, double xi, double xf, int n, int k);
```

Esta función debe calcular la integral usando k cores en paralelo.

Restricción: Ud. debe usar *fork* para crear procesos pesados (no threads). Cada uno de estos procesos calcula la integral de un subintervalo y devuelve el resultado al proceso padre usando un pipe.