CC31A Control 1

2 horas

Septiembre de 2008

Pregunta 1 (punteros y strings)

Parte I

Escriba una función que recibe un string de argumento y retorna el string invertido, usando punteros y no sub-índices (no debe modificar el string original y usar memoria dinámica para el resultado):

```
char *reverse(char *s) {
    char *p, *s2;

    p = (char *)malloc(strlen(s)+1);
    s2 = p+strlen(s);
    *s2-- = 0;
    while(*s)
        *s2-- = *s++;

    return p;
}
```

Parte II

Escriba una función que recibe un string de argumento y lo modifica de forma de invertirlo en el mismo lugar, sin pedir más memoria:

```
void reverse(char *s) {
    char *p;
    char c;


    p = s+strlen(s)-1;
    while(p>s) {
        c = *s;
        *s++ = *p;
        *p-- = c;
```

```
    }
}
```

Parte III

Escriba una función que retorna un substring del argumento, a partir de la posición `i`. Si `i` es positivo, se cuenta desde el comienzo. Si es negativo, desde el final. No pida memoria para el resultado, use el mismo string de origen.

```
char *substring(char *s, int i) {
    if(i >= 0)
        return s+i;
    else
        return s+strlen(s)+i;
}
```

Pregunta 2 (listas)

Queremos implementar strings con listas enlazadas. Definimos el tipo STRING y el string vacío es NULL.

Se les pide escribir las funciones `strcpy` (retorna una copia del string), `strcat` (retorna un nuevo string donde s1 va pegado con s2), `strcmp` (compara dos strings) y `strchr` (retorna un puntero al sub-string que comienza en la letra `c`.

```
typedef struct nodo {   /* cada caracter es un nodo */
    struct nodo *next;
    char c;
} STRING;

STRING *strcpy(STRING *orig) {
   STRING *res, *p;

   if(orig == NULL) return NULL;

   p = (STRING *)malloc(sizeof(STRING));
   if(p==NULL) return NULL;

   p->c = orig->c;
   p->next = Strcpy(orig->next);
   return p;
}

STRING *strcat(STRING *s1, STRING *s2) {
   STRING *p, *res;

   res = Strcpy(s1);
   if(res == NULL) return Strcpy(s2);
   for(p=res; p->next != NULL; p=p->next)
```

```
                ;
   p->next = Strcpy(s2);
   return res;
}

int strcmp(STRING *s1, STRING *s2) {
   if(s1 == NULL && s2 == NULL) return 0;

   if(s1 == NULL) return -1;
   if(s2 == NULL) return 1;

   if(s1->c < s2->c) return -1;
   if(s1->c > s2->c) return 1;
   return Strcmp(s1->next, s2->next);
}

STRING *strchr(STRING *s, char c) {
   if(s == NULL) return NULL;

   if(s->c == c) return s;

   return(Strchr(s->next, c));
}
```

Pregunta 3 (árboles)

Implemente en un ABB la búsqueda de la llave máxima almacenada en el árbol (superior lexicográficamente a todas las otras):

Si el árbol estaba vacío, debe retornar NULL.

```
typedef struct abb {
    char *llave;
    void *val;
    struct abb *left, *right;
} TREE;

char *max_abb(TREE *t) {
   if(t == NULL) return NULL;

   for(; t->right != NULL; t=t->right)
                ;

   return t->llave;
}
```