Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação em Sistemas Computacionais

Teste Global de 2ª Época, Inverno de 2015/2016

Nas questões em que não se indiquem explicitamente outras condições, considere as características do ambiente de referência usado na unidade curricular neste semestre.

1. [3] Sem recorrer a nenhuma função da biblioteca *standard* do C, escreva a função `last_index_of`, que devolve o último índice em `str` onde se encontra o carácter especificado por `search_char`, começando a pesquisa no índice definido por `start`. A função devolve o índice do carácter ou `-1` se o carácter não for encontrado. Considere que `str` contém uma *string* C.

   ```
   int last_index_of(char str[], int start, char search_char);
   ```

2. [2] Considere a representação compacta num `unsigned int` de uma data, onde os 5 *bit* de menor peso representam o dia do mês, os 4 *bit* seguintes representam o mês sendo o ano representado pelos restantes *bit*. Implemente, em linguagem C, as funções `pkdate_get_year` e `pkdate_set_month`. A primeira função devolve o ano da data compacta passada como argumento e a segunda devolve uma nova data, calculada a partir da data e do mês passados como argumentos; se `new_month` especificar um mês inválido, a função deve devolver a data inalterada.

   ```
   typedef unsigned int PkDate;

   unsigned int pkdate_get_year(PkDate date);
   PkDate pkdate_set_month(PkDate date, int new_month);
   ```

3. [3] Considerando as definições abaixo, apresente uma versão da função `compute_score` escrita em *assembly* IA-32.

   ```
   typedef struct score_info ScoreInfo;
   typedef struct score_node ScoreNode;
   struct score_node { ScoreNode *next; ScoreInfo *score; };
   struct score_info { const char *club; char scores[24]; };

   int compute_score(ScoreNode *list, const char *club) {
     for (; list != NULL; list = list->next)
       if (strcmp(list->score->club, club) == 0) {
         int score, i;
         for (score = i = 0; i < 24; i++)
              score += list->score->scores[i];
         return score;
       }
     return -1;
   }
   ```

4. [3] Desenvolva, em *assembly* IA-32, a função `reverse_convert` cuja definição em C se apresenta a seguir.

   ```
   void reverse_convert(void *from[], size_t nitems, size_t size,
                        void (*convert_one)(const void *dst, const void *src), void *to) {
       char *top = (char *)to;
       for (; nitems > 0; top += size)
          (*convert_one)(top, from[--nitems]);
   }
   ```

5. [2] Considere o seguinte conteúdo de dois ficheiro fonte C.

```
/* f1.c */
extern int x;
extern int tab[];
int f(int);
int g(char val) {
  return x += f(tab[val]);
}
int main() {
    return g(2);
}
```

```
/* f2.c */
char x = 'x';
static int tab[] = {1, 2, 3, 5};
static int g(char i) {
    return tab[i & 3];
}
int f(int x) {
    static int a = 56;
    return a *= g(x);
}
```

```
B: bss
D: data
T: text
U: undefined
Maiúscula: global
Minúscula: interna
```

a) [1] Diga quais as tabelas de símbolos associadas dos módulos objecto que resultam da compilação de `f1.c` e `f2.c`. Para cada símbolo, indique o nome, a secção e o respectivo âmbito (i.e., global ou interno).

b) [1] Diga se a ligação entre os módulos `f1.o` e `f2.o` produz erros e, em caso afirmativo, qual o motivo porque ocorrem os erros.

6. [1] Considere uma *cache* com topologia *N-way set associative* com uma capacidade de 4 MiB. Nesta *cache* são utilizados os 14 *bit* de maior peso do endereço para definir a *tag* e os 12 *bit* seguintes para identificar o *set*. Justificando adequadamente, indique o número de *bytes* por linha e o número de vias (*ways*) por *set*.

7. [3] O tipo `Data` representa um item de dados composto por *array* elementos do tipo `double` identificados por uma chave; o *array* é definido pelos campos `length` e `values`, dimensão e endereço, respectivamente. O tipo `DataNode` é um tipo auxiliar destinado a organizar em lista conjuntos de instâncias do tipo `Data`. A função `array_to_list` constrói uma lista simplesmente ligada com as instâncias de `Data` passadas através dos parâmetros `array` e `length`; toda a memória utilizada para construir a lista devolvida por esta função deve ser alocada dinamicamente. A função `free_list` liberta toda a memória alocada dinamicamente pela função `array_to_list`. Implemente, em linguagem C, estas duas funções.

```
typedef struct data { char key[8]; size_t length; double *values; } Data;
typedef struct data_node { struct data_node *next; Data *data; } DataNode;

DataNode *array_to_list(Data array[], size_t length);

void free_list(DataNode *list);
```

8. [3] O código abaixo implementa, em *Java*, uma parte de uma hierarquia de tipos usados em cálculo de áreas de figuras geométricas. Apresente uma implementação, em linguagem C, da classe abstracta `Area` de modo a que seja possível integrar novos tipos na hierarquia sem alterar o código desta classe. Apresente também a implementação da classe `Square`.

```
abstract class Area {
  abstract String type();
  abstract int area();
  final void printAreas(Area[] as) {
    for(Area a : as)
      System.out.println(a.type() + ": " +
                         a.area());
  }
}
```

```
class Square extends Area {
  int side;
  Square(int s) { side = s; }
  String type() { return "square"; }
  int area() { return side * side; }
}

class Circle extends Area {
  int radius;
  Circle(int r) { radius = r; }
  String type() { return "circle"; }
  int area() {
    return (int) (Math.PI * radius *
                  radius); }
}
```

Duração: 2 horas e 30 minutos
ISEL, 10 de Fevereiro de 2016