Gabarito - Lista 1 de AEDS
Listas, Pilhas e Filas

Profa. Virgínia Fernandes Mota

Questão 1

```c
void lst_imprime_rec (Lista *l){
    if (!lst_vazia(l)){
        printf("%d" , l->info);
        lst_imprime_rec(l->prox);
    }
}

Lista *lst_retira_rec (Lista *l, int v){
    if (!lst_vazia(l){
        if (l->info == v){
            Lista *t = l;
            l = l->prox;
            free(t);
        } else {
            l->prox = lst_retira_rec(l->prox, v);
        }
    }
    return l;
}

void lst_libera_rec (Lista *l){
    if (!lst_vazia(l){
        lst_libera_rec(l->prox);
        free(l);
    }
}
```

Questão 2

```c
//Insere na última posição da lista
Lista* lst_insere_ultimo(Lista *l, int i) {
    Lista *novo = (Lista *) malloc (sizeof(Lista));
    Lista *aux = l ;
    if(l == NULL) {
        novo->info = i ;
        novo->prox = NULL ;
        l = novo ;
    }
    else {
        while(aux->prox != NULL)
            aux = aux->prox ;
        novo->info = i ;
        aux->prox = novo ;
        novo->prox = NULL ;
    }
    return l ;
}

Lista* intercala(Lista *l1, Lista *l2) {
    Lista *intercalada = lst_cria();
    do {
        if(l1 != NULL && l2 != NULL) {
            intercalada = lst_insere_ultimo(intercalada, l1->info);
            intercalada = lst_insere_ultimo(intercalada, l2->info);
            l1 = l1->prox ;
            l2 = l2->prox ;
        }
        else if(l1 != NULL) {
            intercalada = lst_insere_ultimo(intercalada, l1->info);
            l1 = l1->prox ;
```

```
        }
        else if(l2 != NULL) {
            intercalada = lst_insere_ultimo(intercalada, l2->info);
            l2 = l2->prox ;
        }
    }
    while(l1 != NULL || l2 != NULL);
    return intercalada ;
}
```

Questão 3

```

#include <stdio.h>
#include <stdlib.h>

typedef struct lista {
    int info;
    struct lista *ant;
    struct lista *prox;
} Lista;

Lista *lst_insord(Lista *l, int v);
void lst_imprime(Lista *l);
Lista *lst_limpar(Lista *l);

int main() {
    Lista *l = NULL;
    l = lst_insord(l, 1);
    l = lst_insord(l, 5);
    l = lst_insord(l, 3);
    l = lst_insord(l, 2);
    l = lst_insord(l, 4);
    lst_imprime(l);
    l = lst_limpar(l);
    printf("\nLimpa\n\n");
    l = lst_insord(l, 5);
    l = lst_insord(l, 1);
    l = lst_insord(l, 3);
    l = lst_insord(l, 2);
    l = lst_insord(l, 4);
    lst_imprime(l);
    return 0;
}

Lista *lst_insord(Lista *l, int v) {
    if(l == NULL) {
        Lista *cel = (Lista *) malloc(sizeof(Lista));
        cel->info = v;
        cel->prox = cel;
        cel->ant  = cel;
        return cel;
    }
    Lista *pos = l, *p = l;
    do
        pos = p;
    while ((p = p->prox) != l && p->info < v);
    Lista *cel = (Lista *) malloc(sizeof(Lista));
    cel->info = v;
    if (pos == l && cel->info < pos->info) {
        cel->prox = pos;
        cel->ant  = pos->ant;
        pos->ant->prox = cel;
        pos->ant  = cel;
        return cel;
    }
    cel->prox = pos->prox;
    pos->prox->ant = cel;
```

```
57      pos->prox = cel;
58      cel->ant  = pos;
59      return l;
60  }
61
62  void lst_imprime(Lista *l) {
63      Lista *p = l;
64      do
65          printf("info %d\n", p->info);
66      while ((p = p->prox) != l);
67  }
68
69  Lista *lst_limpar(Lista *l) {
70      Lista *p = l;
71      do {
72          Lista *rem = p;
73          p = p->prox;
74          free(rem);
75      } while(p != l);
76      return NULL;
77  }
```

Questão 4

```
1
2   /*A função devolve 1 se a cadeia s contém uma sequência
3   bem-formada de parênteses e colchetes e devolve 0 se
4    a sequência está mal formada.*/
5
6   int bemFormada( char s[]){
7       char *pilha; int t;
8       int n, i;
9       n = strlen( s);
10      pilha = (char*)malloc( n * sizeof (char));
11      t = 0;
12      for (i = 0; s[i] != '\0'; i++) {
13          // a pilha está armazenada no vetor pilha[0..t-1]
14          switch (s[i]) {
15              case ')': if (t != 0 && pilha[t-1] == '(' )
16                              t--;
17                        else
18                              return 0;
19                        break;
20              case ']': if (t != 0 && pilha[t-1] == '[')
21                              t--;
22                        else
23                              return 0;
24                        break;
25              default:  pilha[t++] = s[i];
26          }
27      }
28      return t == 0;
29  }
```

Questão 5

```
1   //pilha_push para colocar a cadeia na pilha e depois basta chamar a função
        pilha_ePalindromo.
2   int pilha_ePalindromo (char *cadeia, Pilha *pilha) {
3       int palindromo = 1, i;
4       for (i=0; i<strlen(cadeia); i++)
5           if (cadeia[i] != pilha_pop(pilha)) palindromo = 0;
6       return palindromo;
7   }
```

Questão 6

```
int TTT(int x[], int n){
    if(n == 0)
        return 0 ;
    Pilha *p ;
    p = pilha_cria() ;
    int i, a = 0;
    for(i=0; i<n; i++){
        if(x[i] > 0) {
                pilha_push(p, x[i]);
                a++;
        }
    }
    while(a != 1){
        pilha_push(p, pilha_pop(p) + pilha_pop(p));
        a--;
    }
    return pilha_pop(p);
}
```

Questão 7

```

//A estrutura Fila passa a ser

#define MAX_PLACA 10
#define N 100

typedef struct fila {
    int n;
    int ini;
    char vet[N][MAX_PLACA];
    int idade[N];
} Fila;
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "fila.h"

void menu(int *q);
void pause();

int main() {

    char retira_1[MAX_PLACA];
    int  retira_2;

    Fila *e = fila_cria();

    char str[MAX_PLACA];
    int qq, idade;

    while(1){

    printf("AEDS's Parking -- 24 hours -- only $1 per hour\n\n");
    menu(&qq);
    switch (qq) {

        case 1: { // insere

            int nx;
            printf("Digite a quantidade de carros: ");
            scanf("%d", &nx);

            while (nx--) {
                printf("Digite a placa do carro: ");
                scanf("%s", str);

                printf("Digita a idade do motorista: ");
                scanf("%d", &idade);

                fila_insere(e, str, idade);
            }

            break;
        }
        case 2: { // remove

            printf("Digite a placa do carro: ");
            scanf("%s", str);

            int i = 0, idx = -1;
            for (i = 0; i < e->n; i++)
                if (strcmp(e->vet[((e->ini+i)%N)], str) == 0) {
                    idx = (e->ini+i)%N;
                    break;
                }
```

```
54
55              if (idx == -1)
56                  printf("O carro não encontra-se estacionado.\n");
57              else {
58                  for (i = e->ini; i < idx; i++) {
59                      strcpy(retira_1, fila_retira(e, &retira_2));
60                      fila_insere(e, retira_1, retira_2); // movendo para a entrada
61                  }
62
63                  fila_retira(e, &retira_2); // removendo
64                  printf("O carro foi removido.\n");
65              }
66
67              break;
68          }
69          case 3: { // imprime
70
71              printf("Fila de carros estacionados.\n");
72              fila_imprime(e);
73
74              break;
75          }
76          case 4: {
77              printf("Demolindo o estacionamento...\n");
78              exit(1);
79              break;
80          }
81          default:
82              break;
83      }
84
85      pause();
86      }
87
88      return 0;
89  }
90
91  void menu(int *q) {
92      system("clear");
93
94      printf("1 - Inserir novo carro no estacionamento;\n");
95      printf("2 - Remover carro do estacionamento;\n");
96      printf("3 - Imprimir fila;\n");
97      printf("4 - Sair do programa;\n\n");
98
99      printf("Escolha uma das opções: ");
100     scanf("%d", q);
101
102     system("clear");
103 }
104
105 void pause() {
106     __fpurge(stdin);
107     printf("Aperte ENTER para continuar...");
108     getchar();
109 }
```