Exercícios sobre Pilhas e Filas Estáticas

1) Crie um TAD baseado para manipular uma pilha estática, onde a chave seja um campo do tipo caractere.

```
//Ajustes em relação ao TAD feito em aula, onde a chave era int.

struct item {
    char chave;
}

//adiciona um elemento no fim da Pilha
void empilha(Pilha *p, char chave) {
    if (verificaPilhaCheia(p)) {
        printf("Erro: a pilha está cheia.\n");
        return;
    }
    Item novoItem;
    novoItem.chave = chave;
    p->topo++;
    p->item[p->topo] = novoItem;
}

//imprime a Pilha
void imprimePilha(Pilha *p) {
    int tam = p->topo + 1;
    int i;
    for (i=0; i < tam; i++) {
        printf("Chave: %c\n", p->item[i].chave);
    }
}
```

2) Adaptando o TAD criado no exercício anterior, escreva um programa que solicite ao usuário um texto com até 10 caracteres e realize as seguintes operações usando uma pilha:

- Imprimir o texto na ordem inversa;

```
//imprime a Pilha
void imprimePilha(Pilha *p) {
    int tam = p->topo + 1;
    int i;
    for (i=0; i < tam; i++) {
        printf("Chave: %c\n", p->item[i].chave);
    }
}

#include <stdio.h>
#include "itemPilha.h"

int main(void)
{
    Pilha *p = criaPilhaVazia();
```

```
    char texto[10];
    printf("Informe um texto: ");
    scanf("%[^\n]s", texto);
    for (int i = 0; i < strlen(texto); i++) {
        empilha(p, texto[i]);
    }

    printf("\nDesempilhando: \n");
    for (int i = 0; i < strlen(texto); i++) {
        desempilha(p, 1);
    }

    return 0;
}
```

- Verificar se o texto é um palíndromo, ou seja, se a string é escrita da mesma maneira de frente para trás e de trás para frente. Ignore espaços e pontos.

```
//remove um item qualquer da Pilha
char desempilha(Pilha *p, int imprime) {
    if (verificaPilhaVazia(p)) {
        printf("Erro: a pilha está vazia.\n");
        return;
    }
    char chave = p->item[p->topo].chave;
    if (imprime == 1) {
        printf("%c", chave);
    }
    p->topo--;
    return chave;
}

#include <stdio.h>
#include "itemPilha.h"

int main(void)
{
    Pilha *p = criaPilhaVazia();
    char texto[10];
    printf("Informe um texto: ");
    scanf("%[^\n]s", texto);
    for (int i = 0; i < strlen(texto); i++) {
        empilha(p, texto[i]);
    }

    printf("\nDesempilhando: \n");
    char invertida[10];
    int j = 0;
    while(texto[j] != '\0') {
         invertida[j] = desempilha(p, 1);
         j++;
    }
    invertida[j] = '\0';
    printf("\n%s %s \n", texto, invertida);
    if (strcmp(texto, invertida) == 0) {
        printf("\nEh palindrome");
    } else {
        printf("\nNao eh palindrome");
    }

    return 0;
```

```
}
```

3) Crie uma estrutura de fila baseada em chave do tipo char. Mostre como fica a fila f, inicialmente vazia, após a execução das operações a seguir:

```
enfileira(f,'a');
enfileira(f,'a');
enfileira(f,'b');
enfileira(f,'c');
enfileira(f, espia(f));
enfileira(f, desenfileira(f));
desenfileira(f);
enfileira (f, 'e');
desenfileira(f);
enfileira(f, espia(f));
enfileira(f, 'g');
```

- A função Espia() retorna a chave do item que seria desenfileirado, mas não desenfileira o item;
- A função Desenfileira() retorna a chave do item desenfileirado;
- A função Enfileira() retorna 1 quando consegue enfileirar e 0 quando não consegue.

ItemFilaChar.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "itemFilaExerc.h"

struct item{
    char chave;
};

struct fila {
    Item item[MAXTAM];
    int primeiro;
    int ultimo;
    int tamanho;
};

Fila *criaFilaVazia() {
    Fila *f = malloc(sizeof(Fila));
    f->primeiro = 0;
    f->ultimo = 0;
    f->tamanho = 0;
    return f;
}

int verificaFilaCheia(Fila *f) {
    return f->tamanho == MAXTAM;
}

int enfileira(Fila *f, char chave) {
    if (verificaFilaCheia(f)) {
        printf("Erro: a fila esta cheia.\n");
        return 0;
    }
    Item novoItem;
    novoItem.chave = chave;
    f->item[f->ultimo] = novoItem;
```

```
    f->ultimo = (f->ultimo + 1) % MAXTAM;
    f->tamanho++;
    return 1;
}

void imprimeFila(Fila *f) {
    int t;
    int i = f->primeiro;
    for (t = 0; t < f->tamanho; t++) {
        printf("Chave: %c\n", f->item[i].chave);
        i = (i+1) % MAXTAM;
    }
}

char desenfileira(Fila *f) {
    if (verificaFilaVazia(f)) {
        printf("Erro: a fila está vazia.\n");
        return '_';
    }
    char chaveDes = f->item[f->primeiro].chave;
    f->primeiro = (f->primeiro + 1) % MAXTAM;
    f->tamanho--;
    return chaveDes;
}

int verificaFilaVazia(Fila *f) {
    return f->tamanho == 0;
}

void libera(Fila *f) {
    free(f);
}

char espia(Fila *f) {
    if (verificaFilaVazia(f)) {
        printf("Erro: a fila está vazia.\n");
        return '_';
    }
    return f->item[f->primeiro].chave;
}
```

itemFilaChar.h

```
#define MAXTAM 5

typedef struct item Item;
typedef struct fila Fila;

Fila *criaFilaVazia();
int verificaFilaCheia(Fila *f);
int verificaFilaVazia(Fila *f);
int enfileira(Fila *f, char chave);
void imprimeFila(Fila *f);
char desenfileira(Fila *f);
void liberaFila(Fila *f);
char espia(Fila *f);
Fila *removeDuplicados(Fila *f);
```

usaTADFilaChar.c

```
#include <stdio.h>
#include "itemFilaExerc.h"
```

```
int main(void) {
    Fila *f = criaFilaVazia();
    enfileira(f,'a');
    enfileira(f,'a');
    enfileira(f,'b');
    enfileira(f,'c');
    printf("\nFila apos enfileirar 4 itens. \n");
    imprimeFila(f);
    printf("\nEnfileira o proximo item a ser desenfileirado. \n");
    enfileira(f, espia(f));
    imprimeFila(f);
    printf("\nDesenfileira e enfileira o mesmo item. \n");
    enfileira(f, desenfileira(f));
    imprimeFila(f);
    printf("\nDesenfileirado mais um item. \n");
    desenfileira(f);
    imprimeFila(f);
    printf("\nEnfileira um novo item \'e\'. \n");
    enfileira (f, 'e');
    imprimeFila(f);
    printf("\nDesenfileira um item. \n");
    desenfileira(f);
    imprimeFila(f);
    printf("\nEnfileira o proximo item a ser desenfileirado. \n");
    enfileira(f, espia(f));
    imprimeFila(f);
    printf("\nTenta enfileirar o item \'g\'.\n");
    enfileira(f, 'g');
    printf("\nFila final: \n");
    imprimeFila(f);
    return 0;

}
```

4) Escreva uma nova função para o TAD Fila: a função remove_duplicados(). A função recebe uma fila e remove os itens duplicados (com mesma chave). Você pode utilizar uma fila auxiliar. Exemplo:

- Fila inicial: a, d, d, d, c, f, b, a, h
- Fila com duplicados removidos: a, d, c, f, b, h

```
int buscaChaveFila(Fila *f, char chave) {
    int i = f->primeiro;
    for (int t = 0; t < f->tamanho; t++) {
        if (chave == f->item[i].chave) {
            return 1;
        }
        i = (i+1) % MAXTAM;
    }
    return 0;
}

Fila *removeDuplicados(Fila *f) {
    Fila *f2 = criaFilaVazia();
    int i = f->primeiro;
    for (int t = 0; t < f->tamanho; t++) {
        if (buscaChaveFila(f2, f->item[i].chave) == 0) {//se nao existe
na fila2, insere
            enfileira(f2, f->item[i].chave);
```

```
            }
            i = (i+1) % MAXTAM;
        }
        return f2;
    }
```

5) Escreva uma função que inverte os itens de uma fila utilizando uma pilha auxiliar.

Adaptar e criar as funções abaixo no arquivo itemPilha.c:

```
    int desempilha(Pilha *p) {
        if (verificaPilhaVazia(p)) {
            printf("Erro: a pilha está vazia. \n");
            return;
        }
        p->topo--;
        return p->item[p->topo + 1].chave;
    }

    int retornaTamPilha(Pilha * p) {
        return p->topo + 1;
    }
```

Criar as funções abaixo no arquivo itemPilha.c:

```
Fila *inverteFila(Fila *f) {
    Pilha *p = criaPilhaVazia();
    Fila *invertida = criaFilaVazia();
    int i = f->primeiro;
    for (int t = 0; t < f->tamanho; t++) {
       empilha(p, f->item[i].chave);
       i = (i+1) % MAXTAM;
    }

    int tam = retornaTamPilha(p);
    for (int j = 0; j < tam; j++) {
       enfileira(invertida, desempilha(p));
    }

    libera(p);
    return invertida;
    }
```

No main:

```
    printf("\nInverte fila: \n");
    f = inverteFila(f);
    printf("\nFila final: \n");
    imprimeFila(f);
```