Nome: Gustavo Gomes Dias
13/10/2020

1)Supondo termos um conjunto de dados cuja estrutura seja incrivelmente grande e por isso, demorada de ser copiada de um lugar para o outro. Supondo querermos ordenar esse conjunto de dados segundo algum padrão, ao movermos os dados de um lado para o outro, teríamos uma grande demora, pois obrigatoriamente teríamos que copiar os dados de um lado para o outro. Argumente a veracidade desse texto segundo os conhecimentos de programação em C.

Isso é verdade, pois quanto mais elementos se tem na estrutura, mas trabalho será desencadeado. Isso fica bem mais difícil com C quando ele trabalha com memória, pois é preciso uma série de verificação e conversão.

2)Descreva como uma lista simplesmente encadeada, com todas suas funcionalidades 100% perfeitas, deve ser adaptada para funcionar EXCLUSIVAMENTE como uma fila.

A lista deve se transformar em uma struct que contém dentro dela um ponteiro para ponteiro que aponte para o início e para o fim. O sistema de inserção deve ser modificado para que o elemento que entrar se transforme no primeiro elemento da fila, fazendo com que o ponteiro para ponteiro que aponta para o início aponte para ele e o próximo desse novo elemento seja o que era o primeiro anteriormente. Na fila, caso queira retirar algum elemento, sempre é o último elemento que sai, ou seja, o elemento que estiver setado como o ponteiro para ponteiro que aponte para o fim da fila.

3) Suponha que um cliente impute uma sequência misturada de operações na pilha de PUSH e POP. As operações de PUSH colocam os inteiros 0 ate 9 em ordem na pilha; a POP imprime o valor que for desempilhado. Mostre a ordem correta de empilhar e desempilhar para exibir as sequências abaixo (se for possível, se não for, faça o passo a passo mostrando onde está o erro).

a) 3 5 4 6 9 8 7 2 1 0
b) 6 5 8 4 9 3 2 1 0 7

Letra A
- 1) PUSH 0
- 2) PUSH 1
- 3) PUSH 2
- 4) PUSH 3
- 5) POP (sai o 3)
- 6) IMPRIME 3
- 7) PUSH 4
- 8) PUSH 5
- 9) POP (sai o 5)
- 10) IMPRIME O 5
- 11) POP (sai o 4)
- 12) IMPRIME O 4
- 13) PUSH 6
- 14) POP (sai o 6)
- 15) IMPRIME O 6
- 16) PUSH 7
- 17) PUSH 8
- 18) PUSH 9

- 19) POP (sai o 9)
- 20) IMPRIME O 9
- 21) POP (sai o 8)
- 22) IMPRIME O 8
- 23) POP (sai o 7)
- 24) IMPRIME O 7
- 25) POP (sai o 2)
- 26) IMPRIME O 2
- 27) POP (sai o 1)
- 28) IMPRIME O 1
- 29) POP (sai o 0)
- 30) IMPRIME O 0

Letra B -
A pilha funciona no sistema LIFO (Last in First out), ou seja, o último elemento a entrar é o primeiro elemento a sair e, por conta disso, o último elemento de impressão sempre será o primeiro elemento a entrar. Aqui, como o PUSH é dado em sequência de 0 a 9, sempre deveria ser o 0 o último elemento a ser impresso, logo, não é possível fazer a operação da letra B com as especificações do enunciado, pois para imprimir o 7 por último, seria preciso inserir ele primeiro que o 0.

- ERRO = 1) PUSH 7 -> Deveria ser PUSH 0

1 - PUSH 0 ERRO
2 - PUSH 1
3 - PUSH 2
4 - PUSH 3
5 - PUSH 4
6 - PUSH 5
7 - PUSH 6
8 - POP (sai o 6)
9 - IMPRIME O 6
10 - POP (sai o 5)
11 - IMPRIME O 5
12 - PUSH 7 ERRO (aqui deveria entrar o 7, pois estaria na sequência do PUSH e permitirá a entrada do 8 em seguida só que isso também barraria o POP depois da retirada do 8. pois não saíria o 4 e sim o 7)
13 - PUSH 8
14 - POP (sai o 8)
15 - POP (sai o 7) ERRO (aqui deveria sair o 4)
16 - PUSH 9
17 - POP (sai o 9)
18 - IMPRIME O 9
19 - POP (sai o 3)
20 - IMPRIME O 3
21 - POP (sai o 2)
22 - IMPRIME O 2
23 - POP (sai o 1)
24 - IMPRIME O 1
25 - POP (sai o 0)
26 - IMPRIME O 0

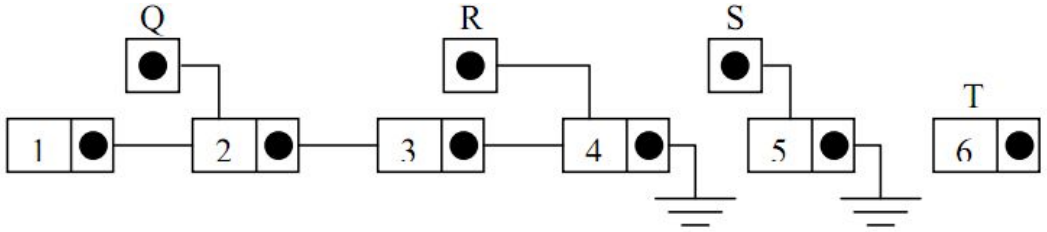
A saída de impressão aqui seria 6->5->8->7->4->9->3->2->1->0

4) Temos um condomínio de 4 apartamentos aonde o dono somente loca um apartamento para um novo locador se o primeiro locador cancelar contrato. Ou seja, se ele locou para 1,2,3,4 ele somente locara pra 5 se o 1 cancelar contrato, da mesma forma somente locara para 6 se o 2 cancelar. Explique qual seria a melhor estrutura a ser feita para atender esse dono e com quais restrições ele deveria ter, supondo ele querer fazer um programa para gerenciar isso.

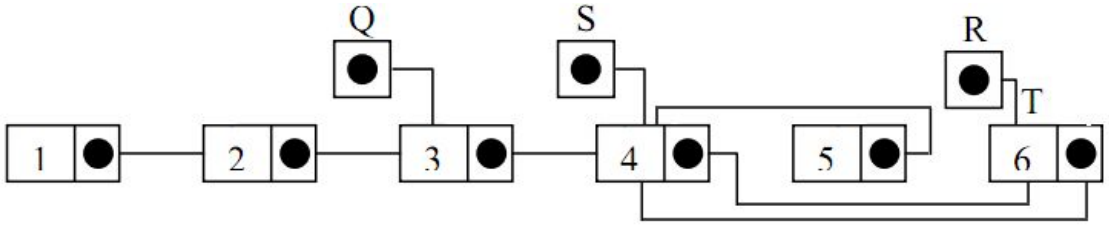
A estrutura ideal aqui seria uma fila, pois ela trabalha no sistema First In First Out, onde o primeiro a entrar é o primeiro a sair, sendo assim, como no exemplo dado, o 1 seria o primeiro a sair e em seguida o 2, isso incorre no que a questão pede, pois caso o 5 for locado, o que teria saído para isso é o 1 e, caso o 6 fosse locado, o que sairia é o 2 (pois depois da saída do 1, ele é o que entrou primeiro) e assim sucessivamente.

Dito isto, aqui se faria uma fila com a restrição de inserir apenas se tiver ocorrido uma chamada da função pop referente a essa fila, ou seja, só seria inserido um novo elemento se fosse retirado o último que entrou.

5)Suponha a seguinte visão da memória abaixo: **(Q, R, S e T são variáveis) as celulas são struct cell { int info; struct cell *prox;};**



Mostre quais são os comandos que deveriam ser feitos para que ela chegasse na situação abaixo SOMENTE usando as 4 variáveis.



Estou imaginando que as variáveis sejam ponteiros cell, sendo assim, Q = cell que contém info igual a 2, por exemplo.
Q = Q->prox;
S->prox = R;
T->prox = R;
R->prox = T;
S = R;
R = T;

6) Qualquer variável que seja tipo: int *, char *, double *, float *, void * ou <qualquer struct aqui dentro> *, ocupam o SEMPRE a mesma quantidade de espaço de memória. Discuta essa frase falando se ela está certa ou errada além de explicar o porquê disso.

O ponteiro contém um endereço de memória do computador e todas as memórias do computador tem o mesmo tamanho, logo, não importa para qual tipo de variável o ponteiro aponta, ele sempre vai ter o tamanho de uma memória do computador.

7) Faça uma função que receba 3 strings (A e B e C) e RETORNE [NÃO É PRA IMPRIMIR] uma NOVA quarta string (D) formada pelos caracteres de A e B e C intercalados. Ex.: Se A='123' e B='abcd' e C='ABCDE', a resposta deve ser '1aA2bB3cCdDE'.

8) Faça uma função que receba uma string como argumento e RETORNE uma NOVA string contendo somente a última palavra da mesma. Exemplo: Se a string digitada for "José da Silva", deverá ser criada a substring "Silva".

char

9) Supondo termos um ponteiro P de um número inteiro que ocupa a posição 100. Esse ponteiro possui memória reservada para 20 números (já preenchidos) que vão de 1 ate 20 em ordem. Qual o valor esperado, supondo que um inteiro ocupe 2 bytes de memória, das equações abaixo?

a. P + 5
b. (*P) + 6
c. *(P + 15)
d. *((int *) (100 + *(P + 5)))
e. (P + 4)

10) Supondo uma fila desenvolvida usando como base uma lista simplesmente encadeada com todos seus métodos de lista já implementados, desenvolva o método
VOID INSERE(FILA * FIL, CELULA *CEL, INT (*PRIORIDADE (CELULA *A, CELULA *B)))
para que ele transforme a fila em uma FILA DE PRIORIDADE onde quem tem maior número de prioridade, deve ser colocado na frente das menores prioridades.
Obs: (*PRIORIDADE(CELULA *A, CELULA *B)) segue o padrão de retorno de STRCMP.

```
void INSERE(FILA * FIL, CELULA *CEL, INT  (*PRIORIDADE (CELULA *A, CELULA *B))){
	CELULA *tmp = get_inicio(FIL);
	CELULA *aux;
	if(get_inicio == NULL){
		set_inicio(FIL, tmp);
		set_fim(FIL, tmp);
	}else{
		if(get_prox(tmp) == NULL){
			if(PRIORIDADE (CEL, tmp)) > 0){
				set_prox(tmp, CEL);
				set_fim(CEL);
			}else{
				set_prox(CEL, tmp);
				set_inicio(FIL, CEL);
				set_fim(tmp);
			}
		}else{
			while(get_prox(tmp) != NULL && PRIORIDADE(CEL, tmp)){
				aux = tmp;
				tmp = get_prox(tmp);
			}
```

```
                        if(get_end(FIL) == tmp){
                                if(PRIORIDADE(CEL, tmp) = 0){
                                        set_prox(aux, CEL);
                                        set_prox(CEL, tmp);

                                }else{
                                        set_prox(tmp, CEL);
                                        set_fim(FIL, CEL);
                                }
                        }
                        if(get_incio(FIL) == tmp){
                                if(PRIORIDADE(CEL, tmp) = 0){
                                        set_prox(CEL, get_prox(tmp));
                                        set_prox(tmp, CEL);
                                }else{
                                        set_prox(CEL, tmp);
                                        set_prox(arg1, CEL);
                                }
                        }
                        set_prox(CEL, tmp);
                        set_prox(aux, CEL);
                }
        }
}
```