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

typedef struct NotabelaHashOpen{
    int chave;
    struct NotabelaHashOpen* proximo;
}NotabelaHashOpen;

typedef struct TabelaHashOpen{
    int tamanho;
    struct NotabelaHashOpen** tabela; //um ponteiro eu crio o vetor o
outro eu guardo um endereço
}TabelaHashOpen;

TabelaHashOpen* criarHash(int tamanho) {
    TabelaHashOpen* H = (TabelaHashOpen*)
malloc(sizeof(TabelaHashOpen));
    if(H != NULL) {
        int i;
        H->tamanho = tamanho;
        H->tabela = (NotabelaHashOpen** ) malloc(tamanho*
sizeof(NotabelaHashOpen*));

        if(H->tabela == NULL) {
            H = NULL;
            printf("Erro na alocacao do item");
            return NULL;
        }
        for (i = 0; i < H->tamanho; i++){
            H->tabela[i] = NULL;
            //(*(*H).tabela[i]).chave = 0;
        }
    }
    return H;
}


int hashFunction(int chave, int tamanho_tabela) {
    return chave % tamanho_tabela;
}

int reHash(int pos_ini,int  i,int tamanho) {
    return (pos_ini + i) % tamanho;
}

void insereHash(TabelaHashOpen* H, int chave){
    int k = hashFunction(chave, H->tamanho);
    int i, nova_posicao;
    for(i = 0; i <H->tamanho; i++) {
        nova_posicao = reHash(k, i, H->tamanho);
        if(H->tabela[nova_posicao] == NULL) {
            NotabelaHashOpen* novo;
            novo = (NotabelaHashOpen*)
malloc(sizeof(NotabelaHashOpen*));
            novo->chave = chave;
            H->tabela[nova_posicao] = novo;
```

```
            return;
        }
    }
    NotabelaHashOpen* aux;
        NotabelaHashOpen* novo =
(NotabelaHashOpen*)malloc(sizeof(NotabelaHashOpen));
        novo->proximo = NULL;
        novo->chave = chave;
        novo->proximo = H->tabela[chave % H->tamanho];
        H->tabela[chave % H->tamanho] = novo;

}

void imprimir(TabelaHashOpen* H, int tamanho) {
    TabelaHashOpen* X = H;
    int i;
    for(i = 0; i < tamanho; i++) {
        printf("posicao %d\n", i);
        if((*(*H).tabela[i]).chave != NULL) {
        printf("%d\n", (*(*X).tabela[i]).chave);
            X = X->tabela[i]->proximo;
        }
    }
}

int main() {
    int tamanho_tabela, opc, chave;
    printf("digite o tamanho da tabela:");
    scanf("%d", &tamanho_tabela);
    TabelaHashOpen* H = criarHash(tamanho_tabela);
    imprimir(H, tamanho_tabela);

    do {
        printf("digite -1 para sair");
        printf("Digite a chave:");
        scanf("%d", &chave);
        insereHash(H, chave);
    }while(chave != -1);
   return 0;

}
```