```c
////=========================================================================
=======================================
//ATIVIDADE PILHA


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

//definindo um tipo abstrato de dado "estrutura_Pilha"
typedef struct{
    //inicializando a variável topo, que indica o tamanho da pilha
    int topo;
    //inicializando o ponteiro que referenciará o vetor com os valodes da
pilha
    int *dados;
    //Informa a capacidade da pilha
    int capacidade;
} estrutura_Pilha;

//Função responsável por cria a pilha
void criaPilha(estrutura_Pilha *p, int c){
    //atribui o parâmetro c ao atributo capacidade da pilha p
    p->capacidade = c;
    //Aloca um espaço na memória com base na capacidade onde será armazenada
a pilha
    p->dados = malloc(p->capacidade * sizeof(estrutura_Pilha));
    //inicializa o atributo topo da pilha p com 0
    p->topo = 0;
}
//Função que empilha um elemento ao final da pilha
void empilha(estrutura_Pilha *p, int A){
    //verifica a capacidade máxima da pilha, em números de elementos
    int n_elementos = (p->capacidade * sizeof(estrutura_Pilha))/sizeof(int);
    //verifica se o número de elementos na lista é menor que sua capacidade
máxima
    if (n_elementos > p->topo){
        //atribui o valor do parâmetro A no final da pilha
        p->dados[p->topo]=A;
        //Acrescenta uma unidade no controle do número de elementos da pilha
        p->topo=p->topo+1;
    }else{
        //caso o número de elementos na lista tenha chegado ao limite imprime
a mensagem abaixo
        printf("Limite da pilha excedido !");
    }

}
//função que empilha elementos até completa a pilha
void empilhaLaco(estrutura_Pilha *p, int A){
    //armazena o tamanho em bytes de um tipo int em uma variável
    int tamanho = sizeof(int);
    //obtêm a capacidade máxima da pilha
    int n_elementos = (p->capacidade * sizeof(estrutura_Pilha))/tamanho;
    //laço responsável por inserir os elementos na pilha
    for(int i = p->topo; i<n_elementos; i++){
        //chama a função empilha para acrescentar o valor A ao final da pilha
        empilha(p, A);
        //imprime o tamanho da pilha ao final de cada repetição
        printf("Tamanho: %d\n", p->topo);
        //Altera o valor de A
```

```
56          A++;
57      }
58  }
59  //função que remove o ultimo elemento da pilha
60  int desempilha (estrutura_Pilha *p){
61      //verifica se a pilha não está vazia
62      if(p->topo ==0){
63          //se a pilha estiver vazia imprime a mensagem abaixo e retorna -1
    (cod. de erro)
64          printf("A pilha está vazia !");
65          return -1;
66      }else{
67          //caso não esteja vazia subtrai uma unidade do topo,
68          //que controla a quantidade de elementos na pilha.
69          //Por fim retorna o elemento que foi removido
70          p->topo=p->topo-1;
71          return p->dados[p->topo];
72      }
73  }
74  //função responsável por imprimir o tamanho da pilha, atributo topo
75  void imprimeTamanho(estrutura_Pilha *p){
76      printf("Tamanho: %d\n", p->topo);
77  }
78  //função responsável por imprimir o valor do ultimo elemento da pilha
79  void imprimeTopo(estrutura_Pilha *p){
80      printf("Topo: %d\n", p->dados[p->topo - 1]);
81  }
82  int main(){
83      //criação de uma variável do tipo abstrato "estrutura_Pilha"
84      estrutura_Pilha pilha1;
85      criaPilha(&pilha1, 17);
86      empilhaLaco(&pilha1, 80);
87      imprimeTamanho(&pilha1);
88      desempilha(&pilha1);
89      imprimeTopo(&pilha1);
90
91      //função responsável por liberar o espaço alocado pela função malloc
92      free(pilha1.dados);
93  }
94
95
96
97
98  //=============================================================================
    ====================================
99  //ATIVIDADE FILA
100
101
102
103 #include<stdio.h>
104 #include<stdlib.h>
105
106 //definindo um tipo abstrato de dado "estrutura_Fila"
107 typedef struct{
108     //variável com a capacidade da Fila
109     int capacidade;
110     //ponteiro para o vetor que armazenará os elementos da Fila
111     int *dados;
112     //variável que representa a posição do primeiro elemento da Fila
113     //na medida em que os primeiros forem retirados
```

```c
    int primeiro;
    //Variável que representa a posição do ultimo elemento da Fila
    //na medida em que forem acresncentados elementos ao final da Fila
    int ultimo;
    //Quantidade total de elementos na Fila
    int quantidade;
} estrutura_Fila;

//função criação da fila
void criarFila(estrutura_Fila *f, int c){

    f->capacidade = c;
    //aloca o espaço na memória para o armazenamento da Fila
    f->dados = malloc(f->capacidade * sizeof(estrutura_Fila));
    //Define primeiro elemento como sendo 0
    f->primeiro = 0;
    //define ultimo elemento inicialmente como 1
    f->ultimo = 1;
    //inicializa quantidade em 0
    f->quantidade = 0;
}

//Insere um elemento ao final da Fila
void inserirFinal(estrutura_Fila *f, int v){
    //Caso o ultimo elemento seja igual a capacidade da Fila
    //ou ainda não exista elemnto na fila é subtraido 1 do atributo
    //ultimo
    if(f->ultimo == f->capacidade-1 || f->quantidade==0){
        f->ultimo--;
    }
    //a posição subsequente ao ultimo valor recebe o parÂmetro v
    f->dados[f->ultimo] = v;
    //o atributo ultimo é deslocado
    f->ultimo++;
    //o atributo quantidade aumenta em uma unidade
    f->quantidade++;
}

//função para remover primeiro elemento
int removerComeco(estrutura_Fila *f){
    //variável temporária armazena o valor que será removido
    int temp = f->dados[f->primeiro];
    //atributo primeiro é deslocado
    f->primeiro++;
    //atributo quantidade é reduzido
    f->quantidade--;
    //a função retorna a variável temporária
    return temp;
}

//laço varre toda a fila imprimindo os elementos
void mostrarFila(estrutura_Fila *f){
    //laço varre fila do primeiro ao ultimo elemento
    for(int cont= f->primeiro; cont < f->ultimo; cont++){
        printf("%d\t", f->dados[cont]);
    }
    printf("\n");
}

int main(){
```

```
174     estrutura_Fila fila1;
175     criarFila(&fila1, 20);
176     inserirFinal(&fila1, 18);
177     inserirFinal(&fila1, 17);
178     inserirFinal(&fila1, 22);
179     mostrarFila(&fila1);
180     removerComeco(&fila1);
181     mostrarFila(&fila1);
182 
183     //função destinada a desalocar os espaço em memória
184     //alocado para a criação da fila
185     free(fila1.dados);
186 }
187 
188 
189 
190 //============================================================================
    ==========================
191 //ATIVIDADE LISTA
192 
193 
194 #include<stdio.h>
195 #include<stdlib.h>
196 
197 //definindo um tipo abstrato de dado "estrutura_Lista"
198 typedef struct{
199     //variável que armazena a quantidade de elementos na lista
200     int ultimo;
201     //ponteiro para o vetor que armazenará os elementos da lista
202     int *dados;
203     //variável com a capacidade da lista
204     int capacidade;
205 } estrutura_Lista;
206 
207 //função de criação da lista
208 void criarLista(estrutura_Lista *l, int c){
209     //atribui o parâmetro c à capacidade da lista
210     l->capacidade = c;
211     //aloca o espaço na memória para o armazenamento da lista
212     l->dados = malloc(l->capacidade * sizeof(estrutura_Lista));
213     //atribui o valor zero ao contador de elementos da lista
214     l->ultimo = 0;
215 }
216 
217 //função responsável por inserir um elemento ao final da lista
218 void insereFinal(estrutura_Lista *l, int A){
219     //atribui o valor do parâmetro A após o ultimo elemento da lsita
220     l->dados[l->ultimo]=A;
221     //Acrescenta uma unidade no contador de elemntos da lista
222     l->ultimo += 1;
223 }
224 
225 //Função responsável por retirar um elemento de uma determinada posição
226 //A posição inicial passará a ter o valor 0
227 void RetiraPos(estrutura_Lista *l, int p){
228     //instância variável auxiliar
229     int Aux;
230     //subtrai uma posição do ponteiro p->dados deslocando o vetor
231     l->dados--;
232     //laço percorre o vetor a partir da posição a ser retirada
```

```c
    for(Aux=p; Aux < l->ultimo; Aux++){
        //cada posição a partir de p recebera o valor de sua subsequente
        l->dados[Aux] = l->dados[Aux+1];
    }
}
//função que insere um valor em uma determinada posição
void inserePos(estrutura_Lista *l, int p, int v){
    //subtrai uma posição do ponteiro p->dados deslocando o vetor
    l->dados--;
    //acrescenta uma unidade no número de elementos
    l->ultimo++;
    //laço para realocar os elementos anteriores à posição p
    //para completar o espaço do deslocamento
    for(int i = 0; i < p; i++){
        //cada posição anterior de p recebera o valor de sua subsequente
        l->dados[i] = l->dados[i+1];
    }
    //atribui o valor do parâmetro v na posição p
    l->dados[p] = v;

}
//função imprime todos os valores do vetor
int imprimeLista(estrutura_Lista *l){

    //laço varre todo o vetor imprimindo cada elemento
    for(int i=0; i<=l->ultimo-1; i++){
        printf("%d\t", l->dados[i]);
    }
    //pula uma linha ao final da impressão do vetor
    printf("\n");
}
//função retorna o número de elementos da lista
int retornaTamanho(estrutura_Lista *l){
    return l->ultimo;
}

int main(){
    estrutura_Lista lista1;
    criarLista(&lista1, 20);
    insereFinal(&lista1, 10);
    insereFinal(&lista1, 20);
    insereFinal(&lista1, 30);
    imprimeLista(&lista1);
    RetiraPos(&lista1, 2);
    imprimeLista(&lista1);
    inserePos(&lista1, 2, 50);
    imprimeLista(&lista1);
    printf("%d\n", lista1.dados[3]);
    printf("\n O tamanho do vetor é: %d\n", retornaTamanho(&lista1));

    free(lista1.dados)
}
```