deque_sequencial_estatico.h

```c
#ifndef DSE_H
#define DSE_H
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

typedef struct{
    int dados[MAX];
    int qtd, inicio, fim;
}Deque;

Deque* criaDeque(){
    Deque* deque;
    deque = (Deque*)malloc(sizeof(Deque));
    if (deque != NULL){
        deque->fim = 0;
        deque->inicio = 0;
        deque->qtd = 0;
    }
    return deque;
}

void destroiDeque(Deque* deque){
    if (deque != NULL)
        free(deque);
}

int tamanhoDeque(Deque* deque){
    if (deque == NULL)
        return -1;
    return deque->qtd;
}

int estaCheio(Deque* deque){
    if (deque == NULL)
        return -1;
    return (deque->qtd == MAX);
}

int estaVazio(Deque* deque){
    if (deque == NULL)
        return -1;
    return (deque->qtd == 0);
}

int insereFim(Deque* deque, int elemento){
    if (deque == NULL)
        return 0;
    if (estaCheio(deque))
        return 0;
    deque->dados[deque->fim] = elemento;
    deque->fim = (deque->fim+1) % MAX;
    deque->qtd++;
    return 1;
}

int insereInicio(Deque* deque, int elemento){
```

```
58      if (deque == NULL)
59          return 0;
60      if (estaCheio(deque))
61          return 0;
62      deque->inicio = (deque->inicio - 1 < 0 ? MAX -1 : deque->inicio-1);
63      deque->dados[deque->inicio] = elemento;
64      deque->qtd++;
65      return 1;
66  }
67
68  int removeFim(Deque* deque){
69      if (deque == NULL)
70          return 0;
71      if (estaVazio(deque))
72          return 0;
73      deque->fim = (deque->fim-1 < 0 ? MAX-1 : deque->fim-1);
74      deque->qtd--;
75      return 1;
76  }
77
78  int removeInicio(Deque* deque){
79      if (deque == NULL)
80          return 0;
81      if (estaVazio(deque))
82          return 0;
83      deque->inicio = (deque->inicio+1)%MAX;
84      deque->qtd--;
85      return 1;
86  }
87
88  int verInicio (Deque* deque, int* elemento){
89      if (deque == NULL)
90          return 0;
91      if (estaVazio(deque))
92          return 0;
93      *elemento = deque->dados[deque->inicio];
94      return 1;
95  }
96
97  int verFim(Deque* deque, int* elemento){
98      if (deque == NULL)
99          return 0;
100     if (estaVazio(deque))
101         return 0;
102     int i = (deque->fim-1 < 0 ? MAX-1 : deque->fim-1);
103     *elemento = deque->dados[i];
104     return 1;
105 }
106
107 void imprime(Deque* deque){
108     if (deque == NULL)
109         return;
110     if (estaVazio(deque)){
111         printf ("Deque vazio!\n");
112         return;
113     }
114     int i = deque->inicio;
115     printf ("Elementos: \n");
116     do{
117         printf ("%d ", deque->dados[i]);
```

```
118         i = (i+1)%MAX;
119     } while (i != deque->fim);
120     printf ("\n");
121 }
122 #endif // DSE_H
```

deque_duplamente_encadeado.h

```c
#ifndef DDE_H
#define DDE_H
#include <stdio.h>
#include <stdlib.h>

typedef struct NO{
    int info;
    struct NO *proximo;
    struct NO *anterior;
} NO;
typedef struct{
    int qtd;
    struct NO *inicio;
    struct NO *fim;
} Deque;

NO *alocarNO()
{
    return (NO *)malloc(sizeof(NO));
}

void liberarNO(NO *q){
    free(q);
}

Deque *criaDeque(){
    Deque *deque;
    deque = (Deque *)malloc(sizeof(Deque));
    if (deque != NULL){
        deque->qtd = 0;
        deque->inicio = NULL;
        deque->fim = NULL;
    }
    return deque;
}

void destroiDeque(Deque *deque){
    if (deque != NULL){
        NO *auxiliar;
        while (deque->inicio != NULL)
        {
            auxiliar = deque->inicio;
            deque->inicio = deque->inicio->proximo;
            liberarNO(auxiliar);
        }
        free(deque);
    }
}

int tamanhoDeque(Deque *deque){
    if (deque == NULL)
        return -1;
    return deque->qtd;
}

int estaVazio(Deque *deque){
    if (deque == NULL)
```

```
58          return -1;
59      return (deque->qtd == 0);
60  }
61
62  int insereInicio(Deque *deque, int elem){
63      if (deque == NULL)
64          return 0;
65      NO *novo = alocarNO();
66      if (novo == NULL)
67          return 0;
68      novo->info = elem;
69      novo->anterior = NULL;
70      if (estaVazio(deque)){
71          novo->proximo = NULL;
72          deque->fim = novo;
73      }
74      else{
75          deque->inicio->anterior = novo;
76          novo->proximo = deque->inicio;
77      }
78      deque->inicio = novo;
79      deque->qtd++;
80      return 1;
81  }
82
83  int insereFim(Deque *deque, int elem){
84      if (deque == NULL)
85          return 0;
86      NO *novo = alocarNO();
87      if (novo == NULL)
88          return 0;
89      novo->info = elem;
90      novo->proximo = NULL;
91      if (estaVazio(deque)){
92          novo->anterior = NULL;
93          deque->inicio = novo;
94      }
95      else{
96          deque->fim->proximo = novo;
97          novo->anterior = deque->fim;
98      }
99      deque->fim = novo;
100     deque->qtd++;
101     return 1;
102 }
103
104 int removeInicio(Deque *deque){
105     if (deque == NULL)
106         return 0;
107     if (estaVazio(deque))
108         return 0;
109     NO *auxiliar = deque->inicio;
110     if (deque->inicio == deque->fim){
111         deque->inicio = deque->fim = NULL;
112     }
113     else{
114         deque->inicio = deque->inicio->proximo;
115         deque->inicio->anterior = NULL;
116     }
117     liberarNO(auxiliar);
```

```
118     deque->qtd--;
119     return 1;
120 }
121 
122 int removeFim(Deque *deque){
123     if (deque == NULL)
124         return 0;
125     if (estaVazio(deque))
126         return 0;
127     NO *auxiliar = deque->fim;
128     if (deque->inicio == deque->fim){
129         deque->inicio = deque->fim = NULL;
130     }
131     else{
132         deque->fim = deque->fim->anterior;
133         deque->inicio->proximo = NULL;
134     }
135     liberarNO(auxiliar);
136     deque->qtd--;
137     return 1;
138 }
139 
140 int verInicio(Deque *deque, int *p){
141     if (deque == NULL)
142         return 0;
143     if (estaVazio(deque))
144         return 0;
145     *p = deque->inicio->info;
146     return 1;
147 }
148 
149 int verFim(Deque *deque, int *p){
150     if (deque == NULL)
151         return 0;
152     if (estaVazio(deque))
153         return 0;
154     *p = deque->fim->info;
155     return 1;
156 }
157 
158 void imprime(Deque *deque){
159     if (deque == NULL)
160         return;
161     if (estaVazio(deque)){
162         printf("Deque Vazio!\n");
163         return;
164     }
165     NO *auxiliar = deque->inicio;
166     printf("Elementos:\n");
167     while (auxiliar != NULL){
168         printf("%d ", auxiliar->info);
169         auxiliar = auxiliar->proximo;
170     }
171     printf("\n");
172 }
173 
174 #endif // DDE_H
```

exercicio1.c

```c
#include <stdio.h>
#include "deque_sequencial_estatico.h"
// #include "deque_duplamente_encadeado.h"
/*
Esse arquivo funciona tanto para o Deque Sequencial Estático, quanto para o Deque Duplamente Encadeado.
É necessário apenas importar a biblioteca que for utilizar
    - deque_sequencial_estatico.h
    ou
    - deque_duplamente_encadeado.h
*/

int main(){
    int option, elemento;
    Deque* deque = NULL;

    do{
        printf ("1- Criar Deque.\n");
        printf ("2- Inserir item no inicio.\n");
        printf ("3- Inserir item no fim.\n");
        printf ("4- Ver o inicio do deque.\n");
        printf ("5- Ver o fim do deque.\n");
        printf ("6- Remover item do fim.\n");
        printf ("7- Remover item do inicio.\n");
        printf ("8- Imprimir deque.\n");
        printf ("9- Destruir deque.\n");
        printf ("10- Sair.\n");

        printf ("Opcao: ");
        scanf ("%d", &option);

        switch (option){
            case 1:
                if (deque != NULL){
                    destroiDeque(deque);
                    printf ("Deque resetado!\n");
                }
                deque = criaDeque();
                break;
            case 2:
                printf ("Digite o numero que deseja adicionar no inicio do deque: \n");
                scanf ("%d", &elemento);
                if (insereInicio(deque, elemento)){
                    printf ("Adicionou (%d) no inicio!\n", elemento);
                } else{
                    printf ("Nao foi possivel adicionar!\n");
                }
                break;
            case 3:
                printf ("Digite o numero que deseja adicionar no fim do deque: \n");
                scanf ("%d", &elemento);
                if (insereFim(deque, elemento)){
                    printf ("Adicionou (%d) no fim!\n", elemento);
                } else{
                    printf ("Nao foi possivel adicionar!\n");
                }
                break;
```

```c
            case 4:
                if (verInicio(deque, &elemento)){
                    printf ("Inicio do Deque = %d\n", elemento);
                } else{
                    printf ("Nao foi possivel ver o inicio.\n");
                }
                break;
            case 5:
                if (verFim(deque, &elemento)){
                    printf ("Fim do Deque = %d\n", elemento);
                } else{
                    printf ("Nao foi possivel ver o Fim.\n");
                }
                break;
            case 6:
                removeFim(deque);
                break;
            case 7:
                removeInicio(deque);
                break;
            case 8:
                imprime(deque);
                break;
            case 9:
                if (deque != NULL)
                    destroiDeque(deque);
                break;
            case 10:
                if (deque != NULL)
                    destroiDeque(deque);
                printf ("Saindo do menu!\n");
                break;
            default:
                printf ("Opcao invalida! Tente de novo.\n");
        }
    } while (option != 10);
    return 0;
}
```

```
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 1
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 2
Digite o numero que deseja adicionar no inicio do deque:
10
Adicionou (10) no inicio!
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 3
Digite o numero que deseja adicionar no fim do deque:
20
Adicionou (20) no fim!
```

```
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 4
Inicio do Deque = 10
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 5
Fim do Deque = 20
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 7
```

```
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 8
Elementos:
20
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 9
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 10
Saindo do menu!
```

```
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 1
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 2
Digite o numero que deseja adicionar no inicio do deque:
10
Adicionou (10) no inicio!
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 3
Digite o numero que deseja adicionar no fim do deque:
20
Adicionou (20) no fim!
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 4
Inicio do Deque = 10
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 5
Fim do Deque = 20
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 7
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 8
Elementos:
20
1- Criar Deque.
2- Inserir item no inicio.
3- Inserir item no fim.
4- Ver o inicio do deque.
5- Ver o fim do deque.
6- Remover item do fim.
7- Remover item do inicio.
8- Imprimir deque.
9- Destruir deque.
10- Sair.
Opcao: 10
Saindo do menu!
```

fila_prioridades_simplesmente_encadeada.h

```c
#ifndef FPSE_H
#define FPSE_H
#include <stdio.h>
#include <stdlib.h>

typedef struct NO{
    int info, prioridade;
    struct NO* proximo;
}NO;
typedef struct NO* FilaPrioridade;

FilaPrioridade* criaFila(){
    FilaPrioridade* filaP;
    filaP = (FilaPrioridade*)malloc(sizeof(FilaPrioridade));
    if (filaP != NULL)
        *filaP = NULL;
    return filaP;
}

NO *alocarNO(){
    return (NO *)malloc(sizeof(NO));
}

void liberarNO(NO *q){
    free(q);
}

int estaVazia(FilaPrioridade *filaP){
    if (filaP == NULL)
        return -1;
    return ((*filaP)==NULL);
}

int tamanho(FilaPrioridade *filaP){
    if (filaP == NULL)
        return -1;
    if (estaVazia(filaP))
        return 0;
    int tamanho = 0;
    NO* auxiliar = *filaP;
    while (auxiliar != NULL){
        auxiliar = auxiliar->proximo;
        tamanho++;
    }
    return tamanho;
}

int inserirPrio(FilaPrioridade* filaP, int elemento, int prioridade){
    if (filaP == NULL)
        return 0;
    NO* novo = alocarNO();
    if (novo == NULL)
        return 0;
    novo->info = elemento;
    novo->prioridade = prioridade;
    if (estaVazia(filaP)){
        novo->proximo = *filaP;
```

```
58          *filaP = novo;
59      } else {
60          NO* auxiliar, *anterior;
61          auxiliar = *filaP;
62          anterior = NULL;
63          while (auxiliar != NULL && auxiliar->prioridade >= novo->prioridade){
64              anterior = auxiliar;
65              auxiliar = auxiliar->proximo;
66          }
67          if (anterior == NULL){
68              novo->proximo = *filaP;
69              *filaP = novo;
70          } else {
71              novo->proximo = anterior->proximo;
72              anterior->proximo = novo;
73          }
74      }
75      return 1;
76  }
77
78  int removeInicio(FilaPrioridade* filaP){
79      if (filaP == NULL)
80          return 0;
81      if (estaVazia(filaP))
82          return 0;
83      NO* auxiliar = *filaP;
84      *filaP = auxiliar->proximo;
85      liberarNO(auxiliar);
86      return 1;
87  }
88
89  int verInicio(FilaPrioridade* filaP, int* elemento, int* prioridade){
90      if (filaP == NULL)
91          return 0;
92      if (estaVazia(filaP))
93          return 0;
94      *elemento = (*filaP)->info;
95      *prioridade = (*filaP).prioridade;
96      return 1;
97  }
98
99  void imprime (FilaPrioridade* filaP){
100     if (filaP == NULL)
101         return;
102     if (estaVazia(filaP)){
103         printf ("Fila de Prioridades Vazia!\n");
104         return;
105     }
106     NO* auxiliar = *filaP;
107     while (auxiliar != NULL){
108         printf ("[%d, %d] ", auxiliar->prioridade, auxiliar->info);
109         auxiliar = auxiliar->proximo;
110     }
111     printf ("\n");
112 }
113
114 void destroiFila(FilaPrioridade* filaP){
115     if (filaP != NULL){
116         NO* auxiliar;
117         while ((*filaP) != NULL){
```

```
118             auxiliar = *filaP;
119             *filaP = (*filaP)->proximo;
120             liberarNO(auxiliar);
121         }
122         free(filaP);
123     }
124 }
125 #endif  //FPSE_H
```

fila_prioridades_heap_binaria.h

```c
#ifndef FPHEAP_H
#define FPHEAP_H
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

typedef struct NO{
    int info, prioridade;
}NO;
typedef struct{
    int qtd;
    NO dados[MAX];
}FilaPrioridade;

FilaPrioridade* criaFila(){
    FilaPrioridade* filaP;
    filaP = (FilaPrioridade*)malloc(sizeof(FilaPrioridade));
    if (filaP != NULL)
        filaP->qtd = 0;
    return filaP;
}

void destroiFila(FilaPrioridade* filaP){
    if (filaP != NULL)
        free(filaP);
}

int tamanho(FilaPrioridade* filaP){
    if (filaP == NULL)
        return -1;
    return filaP->qtd;
}

int estaCheia(FilaPrioridade* filaP){
    if (filaP == NULL)
        return -1;
    return (filaP->qtd == MAX);
}

int estaVazia(FilaPrioridade* filaP){
    if (filaP == NULL)
        return -1;
    return (filaP->qtd = 0);
}

void imprime(FilaPrioridade* filaP){
    if (filaP == NULL)
        return;
    if (estaVazia(filaP)){
        printf ("Fila vazia!\n");
        return;
    }
    printf ("Elementos: \n");
    int i;
    for (i=0; i<filaP->qtd; i++)
        printf ("[%d, %d] (%d) -- ", filaP->dados[i].prioridade, filaP->dados[i].info, i);
    printf ("\n");
```

```
58  }
59
60  void trocaNO(NO* a, NO* b){
61      NO temp;
62      temp.info = a->info;
63      temp.prioridade = a->prioridade;
64      a->info = b->info;
65      a->prioridade = b->prioridade;
66      b->info = temp.info;
67      b->prioridade = temp.prioridade;
68  }
69
70  void ajustaHeapInsere(FilaPrioridade* filaP, int filho){
71      NO temp;
72      int pai = (filho-1)/2;
73      int prioPai = filaP->dados[pai].prioridade;
74      int prioFilho = filaP->dados[filho].prioridade;
75      while(filho > 0 && prioPai < prioFilho){
76          trocaNO(&filaP->dados[filho], &filaP->dados[pai]);
77          filho = pai;
78          pai = (pai-1)/2;
79          prioPai = filaP->dados[pai].prioridade;
80          prioFilho = filaP->dados[filho].prioridade;
81      }
82  }
83
84  int inserirPrio(FilaPrioridade* filaP, int elem, int pri){
85      if(filaP == NULL) return 0;
86      if(estaCheia(filaP)) return 0;
87      filaP->dados[filaP->qtd].info = elem;
88      filaP->dados[filaP->qtd].prioridade = pri;
89      ajustaHeapInsere(filaP, filaP->qtd);
90      filaP->qtd++;
91      return 1;
92  }
93
94  void ajustaHeapRemove(FilaPrioridade* filaP, int pai){
95      NO temp;
96      int filho = 2*pai + 1;
97      while(filho < filaP->qtd){
98          if(filho < filaP->qtd-1)
99              if(filaP->dados[filho].prioridade < filaP->dados[filho+1].prioridade)
100                 filho++;
101
102         if(filaP->dados[pai].prioridade > filaP->dados[filho].prioridade)
103             break;
104
105         trocaNO(&filaP->dados[pai], &filaP->dados[filho]);
106         pai = filho;
107         filho = 2*pai + 1;
108     }
109 }
110
111 int removeInicio(FilaPrioridade* filaP){
112     if(filaP == NULL) return 0;
113     if(estaVazia(filaP)) return 0;
114
115     filaP->qtd--;
116     filaP->dados[0].info = filaP->dados[filaP->qtd].info;
117     filaP->dados[0].prioridade = filaP->dados[filaP->qtd].prioridade;
```

```
118     ajustaHeapRemove(filaP, 0);
119     return 1;
120 }
121 
122 int verInicio(FilaPrioridade* filaP, int* valor, int* pri){
123   if(filaP == NULL)
124     return 0;
125   if(estaVazia(filaP))
126     return 0;
127   *valor = filaP->dados[0].info;
128   *pri = filaP->dados[0].prioridade;
129   return 1;
130 }
131 #endif  //FPHEAP_H
```

```
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 1
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 2
Digite o numero que deseja enfileirar:
10
Prioridade do Elemento:
5
Enfileirou (10)
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 2
Digite o numero que deseja enfileirar:
20
Prioridade do Elemento:
8
Enfileirou (20)
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 3
Inicio: 20 (Prioridade: 8)
```

```
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 5
[8, 20] [5, 10]

1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 6
Tamanho: 2
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 4
Desenfileirou o primeiro elemento.
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 5
[5, 10]
```

```
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 7
Fila destruida!
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 8
Saindo.
```

```
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 1
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 2
Digite o numero que deseja enfileirar:
1
Prioridade do Elemento:
10
Enfileirou (1)
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 2
Digite o numero que deseja enfileirar:
2
Prioridade do Elemento:
5
Enfileirou (2)
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 3
Inicio: 1 (Prioridade: 10)
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 5
Elementos:

1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 6
Tamanho: 0
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 7
Fila destruida!
1- Criar Fila de Prioridades.
2- Inserir item pela prioridade.
3- Ver Inicio.
4- Remover um Item.
5- Imprimir Fila.
6- Mostrar Tamanho.
7- Destruir Fila.
8- Sair.
Opcao: 8
Saindo.
```