```cpp
#include <iostream>
#include <locale.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
using namespace std;



struct funcionario{
    string tarefa;
    int d,m,a;
};
struct gestor{
    string envTaf;
    int d,m,a;
};

struct elemento{
    struct funcionario dados;
    struct gestor dados2;
    struct elemento *prox;

};
typedef struct elemento Elem;
typedef struct elemento* Lista;

Lista* lista_cria(){
    //Lista *li;
    Lista* li = (Lista*) malloc(sizeof(Lista)+1);
    if(li!=NULL)
        *li = NULL;
    return li;
}

int lista_cheia(Lista* li){
    return 0;
}

int lista_vazia(Lista* li){
    if(li == NULL) return -1;
    if(*li == NULL) return 1;
    return 0;
}

void lista_imprime_mat(Lista* li){
    cout << "== TAREFAS DA LISTA ==" << endl;
    Elem *no = *li;
    while(no != NULL){
        cout << no->dados.tarefa << "  ------ Data Limite --> ";
        cout << no->dados.d << "/";
        cout << no->dados.m << "/";
        cout << no->dados.a << endl;
        no = no->prox;
    }
    cout <<"=============FIM=============" << endl;
    cout << endl;
}
void lista_imprime_mat2(Lista* li){
    cout << "== TAREFAS DO GESTOR == " << endl;
    Elem *no = *li;
    while(no != NULL){
        cout << no->dados2.envTaf << "  ------ Data Limite --> ";
        cout << no->dados2.d << "/";
        cout << no->dados2.m << "/";
        cout << no->dados2.a << endl;
        no = no->prox;
    }
    cout <<"=============FIM=============" << endl;
    cout << endl;
}

int lista_insere_inicio(Lista* li, struct funcionario alunoX){
    if(li == NULL) return 0;
    Elem* no = (Elem*) malloc(sizeof(Elem)+1);
    if(no == NULL) return 0;

    no->dados = alunoX;
    no->prox = (*li);
    *li = no;
    return 1;
}


```

```
85  int lista_insere_final(Lista* li, struct funcionario alunoX){
86          if(li == NULL) return 0;
87          Elem* no = (Elem*) malloc(sizeof(Elem)+1);
88          if(no == NULL) return 0;
89
90          no->dados = alunoX;
91          no->prox = NULL;
92
93          if((*li) == NULL)
94              *li = no;
95          else{
96              Elem *aux = *li;
97              while(aux->prox != NULL){
98                  aux = aux->prox;
99              }
100             aux->prox = no;
101         }
102
103      return 1;
104     }
105 int lista_insere_final2(Lista* li, struct gestor al){
106         if(li == NULL) return 0;
107         Elem* no = (Elem*) malloc(sizeof(Elem)+1);
108         if(no == NULL) return 0;
109
110         no->dados2 = al;
111         no->prox = NULL;
112
113         if((*li) == NULL)
114             *li = no;
115         else{
116             Elem *aux = *li;
117             while(aux->prox != NULL){
118                 aux = aux->prox;
119             }
120             aux->prox = no;
121         }
122
123      return 1;
124     }
125  int lista_remove_ordenada(Lista* li, string mat){
126     if(li == NULL) return 0;
127
128     Elem *ant = *li, *no = *li;
129     while(no != NULL && no->dados.tarefa != mat){
130         ant = no;
131         no = no->prox;
132     }
133
134     if(no == NULL) return 0;
135     if(no == *li)
136         *li = no->prox;
137     else
138         ant->prox = no->prox;
139
140     free(no);
141     return 1;
142  }
143
144 int lista_remove_inicio(Lista* li){
145     if(li == NULL) return 0;
146     if(lista_vazia(li)) return 0;
147
148     Elem *no = *li;
149     *li = no->prox;
150     free(no);
151
152     return 1;
153 }
154 int lista_remove_final(Lista* li){
155     if(li == NULL) return 0;
156     if(lista_vazia(li)) return 0;
157
158     Elem *ant = *li, *no = *li;
159     while(no->prox != NULL){
160         ant = no;
161         no = no->prox;
162     }
163
164     if(no == (*li))
165         *li = no->prox;
166     else
167         ant->prox = no->prox;
168     free(no);
```

```
169
170         return 1;
171     }
172
173     int lista_consulta_pos(Lista* li, int pos, struct funcionario *alunoX){
174         if (li ==  NULL || pos <=0) return 0;
175
176         Elem *no = *li;
177         int i = 1;
178         while(no != NULL && i < pos){
179             no = no->prox;
180             i++;
181         }
182
183         if (no ==  NULL)
184             return 0;
185         else{
186             *alunoX = no->dados;
187             return 1;
188         }
189     }
190
191
192     int main(){
193         setlocale(LC_ALL, "Portuguese");
194
195         struct funcionario fu;
196         struct gestor ge;
197
198         Lista *li;
199         Lista *listaF;
200         Lista *listaG;
201         listaF = lista_cria();
202         listaG = lista_cria();
203
204         int escolha;
205         int x=0,j=0,l=0;;
206         int sair = 1;
207
208         while( sair != 2){
209         cout << "========Bem-vindo========" << endl;
210         cout << "=== Escolha entre a lista Funcionario e Gestor ===" << endl << endl;
211         cout << "!!! São 3 dados no total contando a lista Funcionario e a lista Gestor !!!"
    << endl << endl;
212         cout << "Digite 1 - Para Funcionario"<< endl;
213         cout << "Digite 3 - Para Gestor" << endl;
214         cout << "============================" <<endl;
215         cout << "Digite 10 - Para Imprimir a lista Funcionario: " << endl;
216         cout << "Digite 15 - Para imprimir a lista Gestor: " << endl;
217         cout << "============================" << endl;
218         cout << "Digite 2 - Para sair" << endl;
219         cout << "                                        " << endl;
220         cout << "Opção: ";
221         cin >> escolha;
222         cout << endl;
223
224         switch(escolha){
225         case 2:
226             sair = 2;
227             break;
228
229             case 1:
230                 cout << "== LISTA FUNCIONARIO ==" << endl;
231                 cout << "Digite 4 - Para escrever suas tarefas diarias na lista" << endl;
232                 cout << "Digite 10 - Para acessar tarefas escritas pelo GESTOR" << endl;
233                 cout << "Digite 2 - Para sair" << endl;
234                 cin >> j;
235                 cout << endl;
236                     while(j ==  4){
237                             cout << "== LISTA DE TAREFAS ==" << endl;
238                             cin.ignore( 1000, '\n' );
239                             getline(cin,(fu.tarefa));
240
241                             cout << "Escreva a data LIMITE da entrega" << endl;
242                             cout << "Dia: ";
243                             cin >> fu.d;
244                             cout << "Mês: ";
245                             cin >> fu.m;
246                             cout << "Ano: ";
247                             cin >> fu.a;
248                             lista_insere_final(listaF, fu);
249
250                             cout << endl;
251                             lista_imprime_mat(listaF);
```

```
252                     cout << "Voce deseja inserir outra tarefa ?"; cout << "== MAXIMO DE 3
    TAREFAS POR LISTA ==" << endl;
253                     cout << "Digite 4 - Para inserir novamente" << endl;
254                     cout << "Digite 2 - Para sair da lista" << endl;
255                     cout << "Digite 6 - Para remover uma tarefa" << endl;
256                     cin.ignore( 1000, '\n' );
257                     cin >> j;
258                     cout << endl;
259                  }if( j == 10 ){
260                     cout << "== LISTA DO GESTOR ==" << endl;
261                     cout << "==========================" << endl;
262                     lista_imprime_mat2(listaG);
263                     cout << "------Lista Inserida------" << endl << endl;
264                     system("pause");
265                     break;
266                  }
267                  if(j == 6){
268
269                     lista_remove_inicio(li);
270                     cout << "=========================================" << endl;
271                     cout << "Escreva 1- para remover a primeira tarefa" << endl;
272                     cout << "Escreva 2- para remover a ultima tarefa" << endl;
273                     cin >> l;
274                     if(l == 1){
275                         lista_remove_inicio(listaF);
276                         lista_imprime_mat(listaF);
277                     }
278                     if(l == 2){
279                         lista_remove_final(listaF);
280                         lista_imprime_mat(listaF);
281                     }
282                  }
283              break;
284
285          case 3:
286              cout << "== LISTA DO GESTOR ==" << endl;
287              cout << "Digite 4 - Para escrever a tarefa para o funcionario" << endl;
288              cout << "Digite 15 - Para acessar as tarefas enviadas para o funcionario" <<
    endl;
289              cout << "Digite 2 - Para sair" << endl;
290              cin >> j;
291                  while(j == 4){
292                      cout << endl;
293                      cout << "== LISTA DO GESTOR ==" << endl;
294                      cin.ignore( 1000, '\n' );
295                      getline(cin,(ge.envTaf));
296
297                      cout << "Escreva a data LIMITE da entrega" << endl;
298                      cout << "Dia: ";
299                      cin >> ge.d;
300                      cout << "Mês: ";
301                      cin >> ge.m;
302                      cout << "Ano: ";
303                      cin >> ge.a;
304                      lista_insere_final2(listaG, ge);
305
306                      cout << endl;
307                      lista_imprime_mat2(listaG);
308                      cout << "Voce deseja inserir outra tarefa ?"; cout << "== MAXIMO DE 3
    TAREFAS POR LISTA ==" << endl;
309                      cout << "Digite 4 - Para inserir novamente" << endl;
310                      cout << "Digite 6 - Para remover uma tarefa" << endl;
311                      cout << "Digite 2 - Para sair da lista" << endl;
312
313                      cin >> j;
314                      cout << endl;
315                  } if(j == 15){
316                      cout << "== TAREFAS DO FUNCIONARIO ==" << endl;
317                      cout << "____________________________" << endl;
318                      lista_imprime_mat(listaF);
319                      system("pause");
320                      break;
321                   }
322
323
324                  if(j == 6){
325
326                  lista_remove_inicio(li);
327                  cout << "=========================================" << endl;
328                  cout << "Escreva 1- para remover a primeira tarefa" << endl;
329                  cout << "Escreva 2- para remover a ultima tarefa" << endl;
330                  cin >> l;
331                  if(l == 1){
332                      lista_remove_inicio(listaG);
```

```
333                     lista_imprime_mat(listaG);
334                 }
335                 if(l == 2){
336                     lista_remove_final(listaG);
337                     lista_imprime_mat(listaG);
338                 }
339                 system("pause");
340                 break;
341                 }
342             break;
343         case 10:
344              cout << "== LISTA DO FUNCIONARIO ==" << endl;
345                 cout << "==========================" << endl;
346                 lista_imprime_mat(listaF);
347                 system("pause");
348                 break;
349         case 15:
350             cout << "== LISTA DO GESTOR ==" << endl;
351                 cout << "_____________________________" << endl;
352                 lista_imprime_mat(listaG);
353                 system("pause");
354                 break;
355         }
356     }
357     return 0;
358 }
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
```