```c

#include <stdio.h>
#include "prjstud.h"

struct anonApplication  *sortByUniversity(struct anonApplication *head)
{
    struct anonApplication* ptr1, * ptr2, *temp1, * temp2, * temp3;
    ptr1 = head;

    while (ptr1 != NULL)
    {
        ptr2 = ptr1->nextApp;
        while (ptr2 != NULL)
        {
            if((ptr1->university[0] < ptr2->university[0])!=0) //university name is already in alphabetical order
            {
                ptr2 = ptr2->nextApp;
            }
            else if((ptr1->university[0] > ptr2->university[0])!=0) //university name needs to be swapped
            {
                if (ptr1 == head) //beginning of list
                {
                    temp2 = head;
                    while (temp2->nextApp != ptr2)
                    {
                        temp2 = temp2->nextApp;
                    }
                    temp2->nextApp = ptr1;
                    head = ptr2;
                    temp1 = ptr2->nextApp;
                    ptr2->nextApp = ptr1->nextApp;
                    ptr1->nextApp = temp1;

                }
                else //in list
                {
                    temp2 = head;
                    temp3 = head;
                    while (temp2->nextApp != ptr2)
                    {
                        temp2 = temp2->nextApp;
                    }
                    while (temp3->nextApp != ptr1)
                    {
                        temp3 = temp3->nextApp;
                    }
                    temp3->nextApp = ptr2;
                    temp2->nextApp = ptr1;
                    temp1 = ptr2->nextApp;
                    ptr2->nextApp = ptr1->nextApp;
```

```
51                      ptr1->nextApp = temp1;
52
53                  }
54                  ptr1 = ptr2;
55                  ptr2 = ptr2->nextApp;
56              }
57              else
58              {
59                  if ((ptr1->university[1] <= ptr2->university[1]) != 0) //
                      university name is already in alphabetical order
60                  {
61                      ptr2 = ptr2->nextApp;
62                  }
63                  else //university name needs to be swapped
64                  {
65                      if (ptr1 == head) //beginning of list
66                      {
67                          temp2 = head;
68                          while (temp2->nextApp != ptr2)
69                          {
70                              temp2 = temp2->nextApp;
71                          }
72                          temp2->nextApp = ptr1;
73                          head = ptr2;
74                          temp1 = ptr2->nextApp;
75                          ptr2->nextApp = ptr1->nextApp;
76                          ptr1->nextApp = temp1;
77
78                      }
79                      else //in list
80                      {
81                          temp2 = head;
82                          temp3 = head;
83                          while (temp2->nextApp != ptr2)
84                          {
85                              temp2 = temp2->nextApp;
86                          }
87                          while (temp3->nextApp != ptr1)
88                          {
89                              temp3 = temp3->nextApp;
90                          }
91                          temp3->nextApp = ptr2;
92                          temp2->nextApp = ptr1;
93                          temp1 = ptr2->nextApp;
94                          ptr2->nextApp = ptr1->nextApp;
95                          ptr1->nextApp = temp1;
96
97                      }
98                      ptr1 = ptr2;
99                      ptr2 = ptr2->nextApp;
100                 }
101
```

```
102             }
103         }
104         ptr1 = ptr1->nextApp;
105     }
106     return head;
107 }
108
109
110 struct anonApplication *deleteApplication  (struct anonApplication *head, struct
      anonApplication *to_be_gone)
111 {
112     struct anonApplication* i = head;
113
114     if (i == to_be_gone)
115     {
116         head = i->nextApp;
117     }
118     else
119     {
120         while (i->nextApp != to_be_gone)
121         {
122             i = i->nextApp;
123         }
124         i->nextApp = i->nextApp->nextApp;
125     }
126
127     return head;
128 }
129
130 struct anonApplication *processDay(struct anonApplication* head, int *clockTime)
131 {
132     struct anonApplication* hold = head;
133     struct anonApplication* NewApplication(int clocktime);
134     struct anonApplication* ptr;
135
136     *clockTime=*clockTime+1;
137     while (hold != NULL)
138     {
139         if (hold->timeIn >= 5)
140         {
141             head=deleteApplication(head, hold);
142         }
143         hold = hold->nextApp;
144     }
145
146     head = sortByAvgSkill(head);
147     hold = head;
148     while (hold != NULL)
149     {
150         if (strcmp(hold->positionApplied, "systems engineer")==0)
151         {
152             printf("appID %d has been hired for the position is for systems
```

```
                engineer\n", hold->appID);
153             break;
154         }
155         else
156         {
157             hold = hold->nextApp;
158         }
159     }
160     if (hold != NULL)
161     {
162         head = deleteApplication(head, hold);
163     }
164
165     head = sortByAvgSkill(head);
166     hold = head;
167     while (hold != NULL)
168     {
169         if (strcmp(hold->positionApplied, "software engineer") == 0)
170         {
171             printf("appID %d has been hired for the position is for software
                engineer\n", hold->appID);
172             break;
173         }
174         else
175         {
176             hold = hold->nextApp;
177         }
178     }
179     if (hold != NULL)
180     {
181         head = deleteApplication(head, hold);
182     }
183
184     head = sortByAvgSkill(head);
185     hold = head;
186     while (hold != NULL)
187     {
188         if (strcmp(hold->positionApplied, "controls engineer") == 0)
189         {
190             printf("appID %d has been hired for the position is for controls
                engineer\n", hold->appID);
191             break;
192         }
193         else
194         {
195             hold = hold->nextApp;
196         }
197     }
198     if (hold != NULL)
199     {
200         head = deleteApplication(head, hold);
201     }
```

```
202
203     head = sortByAvgSkill(head);
204     hold = head;
205     while (hold != NULL)
206     {
207         if (strcmp(hold->positionApplied, "engineering management") == 0)
208         {
209             printf("appID %d has been hired for the position is for engineering
                  management\n", hold->appID);
210             break;
211         }
212         else
213         {
214             hold = hold->nextApp;
215         }
216     }
217     if (hold != NULL)
218     {
219         head = deleteApplication(head, hold);
220     }
221
222     head = sortByAvgSkill(head);
223     hold = head;
224     while (hold != NULL)
225     {
226         hold->timeIn = hold->timeIn + 1;
227         hold = hold->nextApp;
228     }
229
230     ptr = NewApplications(*clockTime);
231     head = sortByAvgSkill(head);
232     hold = head;
233     while (hold->nextApp != NULL)
234     {
235         hold = hold->nextApp;
236     }
237     hold->nextApp = ptr;
238     head = sortByAvgSkill(head);
239
240     return head;
241 }
242
243 void printFormatted (struct anonApplication * head){
244     struct anonApplication* temp = head;
245     printf("appID  appDate  timeIn  ProgSkill  CircDesign  ProjManage    GPA
          University       positionApplied
        \n=============================================================================
        ========================================\n");
246     while (temp != NULL)
247     {
248         printf("%d      %d        %d       %d          %d           %d        %f
                %s        %s\n", temp->appID, temp->appDate, temp->timeIn, temp-
```

```
            >skillLevelProgramming, temp->skillLevelCircDesign, temp-
            >skillLevelPrjManage, temp->collegeGPA, temp->university, temp-
            >positionApplied);
249         temp = temp->nextApp;
250     }
251 }
252
253 void analyzeApplicantList (struct anonApplication * head)
254 {
255     struct anonApplication *temp = head;
256     double avgGPA[4] = { 0 };
257     double avgCD[4] = { 0 };
258     double avgPM[4] = { 0 };
259     double appnum[4] = { 0 };
260     double avgPROG[4] = { 0 };
261
262     while (temp != NULL)
263     {
264
265         if (strcmp(temp->positionApplied, "systems engineer") == 0)
266         {
267             appnum[0] += 1;
268             avgGPA[0] += temp->collegeGPA;
269             avgCD[0] += temp->skillLevelCircDesign;
270             avgPM[0] += temp->skillLevelPrjManage;
271             avgPROG[0] += temp->skillLevelProgramming;
272             temp = temp->nextApp;
273         }
274         else if (strcmp(temp->positionApplied, "software engineer") == 0)
275         {
276             appnum[1] += 1;
277             avgGPA[1] += temp->collegeGPA;
278             avgCD[1] += temp->skillLevelCircDesign;
279             avgPM[1] += temp->skillLevelPrjManage;
280             avgPROG[1] += temp->skillLevelProgramming;
281             temp = temp->nextApp;
282         }
283         else if (strcmp(temp->positionApplied, "controls engineer") == 0)
284         {
285             appnum[2] += 1;
286             avgGPA[2] += temp->collegeGPA;
287             avgCD[2] += temp->skillLevelCircDesign;
288             avgPM[2] += temp->skillLevelPrjManage;
289             avgPROG[2] += temp->skillLevelProgramming;
290             temp = temp->nextApp;
291         }
292         else
293         {
294             appnum[3] += 1;
295             avgGPA[3] += temp->collegeGPA;
296             avgCD[3] += temp->skillLevelCircDesign;
297             avgPM[3] += temp->skillLevelPrjManage;
```

```
298             avgPROG[3] += temp->skillLevelProgramming;
299             temp = temp->nextApp;
300         }
301     }
302     avgGPA[0] = avgGPA[0] / appnum[0];
303     avgCD[0] = avgCD[0] / appnum[0];
304     avgPM[0] = avgPM[0] / appnum[0];
305     avgPROG[0] = avgPROG[0] / appnum[0];
306
307     avgGPA[1] = avgGPA[1] / appnum[1];
308     avgCD[1] = avgCD[1] / appnum[1];
309     avgPM[1] = avgPM[1] / appnum[1];
310     avgPROG[1] = avgPROG[1] / appnum[1];
311
312     avgGPA[2] = avgGPA[2] / appnum[2];
313     avgCD[2] = avgCD[2] / appnum[2];
314     avgPM[2] = avgPM[2] / appnum[2];
315     avgPROG[2] = avgPROG[2] / appnum[2];
316
317     avgGPA[3] = avgGPA[3] / appnum[3];
318     avgCD[3] = avgCD[3] / appnum[3];
319     avgPM[3] = avgPM[3] / appnum[3];
320     avgPROG[3] = avgPROG[3] / appnum[3];
321
322     printf("System Engineering Info\n\nAverage GPA:%f\nAverage Circuit Design
        Skill Level:%f\nAverage Project Management Skill Level:%f\nAverage
        Programming Skill Level:%f\n\n", avgGPA[0], avgCD[0], avgPM[0], avgPROG
        [0]);
323     printf("Software Engineering Info\n\nAverage GPA:%f\nAverage Circuit Design
        Skill Level:%f\nAverage Project Management Skill Level:%f\nAverage
        Programming Skill Level:%f\n\n", avgGPA[1], avgCD[1], avgPM[1], avgPROG
        [1]);
324     printf("Controls Engineering Info\n\nAverage GPA:%f\nAverage Circuit Design
        Skill Level:%f\nAverage Project Management Skill Level:%f\nAverage
        Programming Skill Level:%f\n\n", avgGPA[2], avgCD[2], avgPM[2], avgPROG
        [2]);
325     printf("Engineering Management Info\n\nAverage GPA:%f\nAverage Circuit Design
         Skill Level:%f\nAverage Project Management Skill Level:%f\nAverage
        Programming Skill Level:%f\n\n", avgGPA[3], avgCD[3], avgPM[3], avgPROG
        [3]);
326 }
327
328 void terminateAndWrite(struct anonApplication * head)
329 {
330     FILE* fptr = fopen("output.txt", "w");
331     struct anonApplication* temp = head;
332     fprintf(fptr, "appID  appDate  timeIn  ProgSkill  CircDesign  ProjManage
        GPA       University       positionApplied
        \n=============================================================================
        ========================================\n");
333     while (temp != NULL)
334     {
```

```
335         fprintf(fptr, "%d      %d        %d       %d          %d           %d     ↵
                  %f     %s        %s\n", temp->appID, temp->appDate, temp->timeIn,  ↵
              temp->skillLevelProgramming, temp->skillLevelCircDesign, temp-        ↵
              >skillLevelPrjManage, temp->collegeGPA, temp->university, temp-        ↵
              >positionApplied);
336         temp = temp->nextApp;
337     }
338     fclose(fptr);
339     printf("\n\n\nbye!\n\n\n");
340     exit(0);
341 }
342
```