```c
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

#include "H_Dictionary.h"
#include "H_Model.h"

/*------------------------------------------------------------------------------------------*/
/*------------------------------------ Function Prototypes ---------------------------------*/
/*------------------------------------------------------------------------------------------*/

/*------------------------------------- AUTHORIZATION --------------------------------------*/

bool login(void);

/*---------------------------------- Dictionary Function Headers ---------------------------*/

void initDictionary(dPtr *D);
ElemPos hash(ID id);
int getNewID(char DictionaryType[], Dictionary D);
bool pullDictionaries(Dictionary *D);
bool pushDictionaries(Dictionary D);

/*---------------------------------- Attendance Function Headers ---------------------------*/

Model_Attendance *searchAttendance(Dictionary D, ID employeeID, char period[]);
Model_Attendance createAttendance(Dictionary D, int empID, char period[]);
bool insertAttendance(Dictionary *D, Model_Attendance data);
bool deleteAttendance(Dictionary *D, int empID, char period[]);
bool setPresent(int employeeID, Dictionary *D, char period[]);
bool setLeave(int employeeID, Dictionary *D, char period[]);
bool setAbsent(int employeeID, Dictionary *D, char period[]);
bool setOvertime(int employeeID, Dictionary *D, char period[]);
void displayAttendance(Dictionary *D, int employeeID, char period[]);
void displayAllAttendance(Dictionary D, char period[]);

/*---------------------------------- Bonus Function Headers --------------------------------*/

Model_Bonus *searchBonus(Dictionary D, ID bonusID, char period[]);
Model_Bonus createBonus(Dictionary D, int employeeID, char period[]);
bool insertBonus(Dictionary *D, Model_Bonus data);
bool deleteBonus(Dictionary *D, ID employeeID, char period[]);
void displayBonus(int bonusID, Dictionary *D, char period[]);
void displayAllBonus(Dictionary D, char period[]);

/*------------------------------- Issue Salary Function Headers ----------------------------*/

Model_IssueSalary *searchIssueSalary(Dictionary D, ID userID, char period[]);
```

```
51 Model_IssueSalary createIssueSalary(Dictionary D, int employeeID, double balance,
   double pagibig, char period[]);
52 bool insertIssueSalary(Dictionary *D, Model_IssueSalary data);
53 bool deleteIssueSalary(Dictionary *D, ID issueID, char period[]);
54 bool issueSalary(Dictionary *D, int empID, char period[]);
55 double calculateTax(double basicIncome, double taxableIncome, double
   *pagibigDeposit, double *pagibig);
56 void displayIssueSalary(Dictionary *D, ID empID, char period[]);
57 void displayAllSalary(Dictionary D, char period[]);
58
59 /*------------------------------ Job Information Function Headers ----------------
   ---------------*/
60
61 Model_JobInformation *searchJobInformation(Dictionary D, ID employeeID);
62 Model_JobInformation createJobInformation(Dictionary D, ID employeeID);
63 bool insertJobInformation(Dictionary *D, Model_JobInformation data);
64 bool deleteJobInformation(Dictionary *D, ID employeeID);
65 void displayJobInformation(ID employeeID, Dictionary *D);
66 void displayAllJobInformation(Dictionary D);
67
68 /*------------------------------------ User Function Headers ---------------------
   ---------------*/
69
70 Model_User *searchUser(Dictionary D, ID userID);
71 Model_User createUserInformation(Dictionary D);
72 bool insertUser(Dictionary *D, Model_User data);
73 bool deleteUser(Dictionary *D, ID userID);
74 void displayUserInformation(ID userID, Dictionary *D);
75 void displayAllUser(Dictionary D);
76
77 /*------------------------------------------- Default ----------------------------
   ---------------*/
78
79 void insertDefault(Dictionary *D);
80
81 /*-------------------------------------------- Debug ----------------------------
   ---------------*/
82
83 void debugAttendance(Dictionary D);
84 void debugBonus(Dictionary D);
85 void debugSalary(Dictionary D);
86 void debugJobInformation(Dictionary D);
87 void debugUser(Dictionary D);
88 void debugAll(Dictionary D);
89
90 /*---------------------------------------------- UI ------------------------------
   ---------------*/
91
92 void header(void);
93 void invalidInput(void);
94
95 /*------------------------------------------- DEBUG ----------------------------
   ---------------*/
96
97 void displayDictionariesCount(Dictionary D);
98
99 /*------------------------------------------------------------------------------
   ---------------*/
100 /*---------------------------------- Function Definitions  ---------------------
   ---------------*/
```

```c
/*-------------------------------------------------------------------------------------------*/

/*------------------------------ Start of Debug All Controller ------------------------------*/

void debugAll(Dictionary D)
{
    debugAttendance(D);
    debugBonus(D);
    debugSalary(D);
    debugJobInformation(D);
    debugUser(D);
}

/*------------------------------- End of Debug All Controller -------------------------------*/

/*----------------------------- Start of Authorization Controller -----------------------------*/

bool login(void)
{
    char USERNAME[MD_MAX] = "admin", PASSWORD[MD_MAX] = "admin", user[MD_MAX], pass[MD_MAX];

    printf(" WELCOME TO EDWIN'S BEST PAYROLL SYSTEM\n\n");
    printf(" Login to System\n\n");
    printf(" Username: ");
    scanf("%s", &user);
    fflush(stdin);
    printf(" Password: ");
    scanf("%s", &pass);
    fflush(stdin);

    if (strcmp(USERNAME, user) == 0 && strcmp(PASSWORD, pass) == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}

/*------------------------------ End of Authorization Controller ------------------------------*/

/*------------------------------- Start of Dictionary Controller ----------------------------*/

void initDictionary(dPtr *D)
{
    *D = (dPtr)calloc(1, sizeof(Dictionary));
}

ElemPos hash(ID id)
{
    return id % 10;
}
```

```
154
155 int getNewID(char DictionaryType[], Dictionary D)
156 {
157     int lastID;
158
159     if (strcmp(DictionaryType, "Attendance") == 0)
160     {
161         lastID = D.count[0];
162     }
163     else if (strcmp(DictionaryType, "Bonus") == 0)
164     {
165         lastID = D.count[1];
166     }
167     else if (strcmp(DictionaryType, "IssueSalary") == 0)
168     {
169         lastID = D.count[2];
170     }
171     else if (strcmp(DictionaryType, "JobInformation") == 0)
172     {
173         lastID = D.count[3];
174     }
175     else if (strcmp(DictionaryType, "User") == 0)
176     {
177         lastID = D.count[4];
178     }
179     else
180     {
181         printf(" Invalid Dictionary Type\n");
182         return -1;
183     }
184
185     return lastID + 1;
186 }
187
188 bool pullDictionaries(Dictionary *D)
189 {
190     // Attendance Information
191     FILE *fp = fopen("Emp_Attendance.bin", "rb");
192     Model_Attendance attendance;
193
194     if (fp)
195     {
196         while (fread(&attendance, sizeof(Model_Attendance), 1, fp))
197         {
198             printf(" Attendance inserting to dictionary...\n");
199             insertAttendance(D, attendance);
200         }
201
202         fclose(fp);
203     }
204     else
205     {
206         return false;
207     }
208
209     // Bonus Information
210     fp = fopen("Emp_Bonus.bin", "rb");
211     Model_Bonus bonus;
212
213     if (fp)
```

```
    {
        while (fread(&bonus, sizeof(Model_Bonus), 1, fp))
        {
            insertBonus(D, bonus);
        }
        fclose(fp);
    }
    else
    {
        return false;
    }

    // Salary Information
    fp = fopen("Emp_IssueSalary.bin", "rb");
    Model_IssueSalary salary;

    if (fp)
    {
        while (fread(&salary, sizeof(Model_IssueSalary), 1, fp))
        {
            insertIssueSalary(D, salary);
        }
        fclose(fp);
    }
    else
    {
        return false;
    }

    // Job Information
    fp = fopen("Emp_JobInformation.bin", "rb");
    Model_JobInformation jobinfo;

    if (fp)
    {
        while (fread(&jobinfo, sizeof(Model_JobInformation), 1, fp))
        {
            insertJobInformation(D, jobinfo);
        }
        fclose(fp);
    }
    else
    {
        return false;
    }

    // Emp Information
    fp = fopen("Emp_Information.bin", "rb");
    Model_User employee;

    if (fp)
    {
        while (fread(&employee, sizeof(Model_User), 1, fp))
        {
            insertUser(D, employee);
        }
        fclose(fp);
    }
    else
    {
```

```
274         return false;
275     }
276 
277     return true;
278 }
279 
280 bool pushUserD(Dictionary D)
281 {
282     PSU UTrav;
283     int i, count;
284     FILE *fp;
285 
286     // User Information
287     fp = fopen("Emp_Information.bin", "wb");
288     if (fp)
289     {
290         for (i = 0; i < DICT_SIZE; i++)
291         {
292             for (UTrav = D.UserD[i]; UTrav != NULL; UTrav = UTrav->next)
293             {
294                 fwrite(&UTrav->data, sizeof(Model_User), 1, fp);
295             }
296         }
297 
298         // fseek(fp, 0L, SEEK_END);
299         // if (ftell(fp) == -1)
300         // {
301         //     count = 0;
302         // }
303         // else
304         // {
305         //     count = ftell(fp) / sizeof(Model_User);
306         // }
307 
308         // if (count != D.count[4])
309         // {
310         //     return false;
311         // }
312 
313         fclose(fp);
314     }
315     else
316     {
317         printf(" \n ERROR: Failed to create file.");
318         return false;
319     }
320 
321     return true;
322 }
323 
324 bool pushJobInformationD(Dictionary D)
325 {
326     PSJI JITrav;
327     int i, count;
328     FILE *fp;
329 
330     // Job Information
331     fp = fopen("Emp_JobInformation.bin", "wb");
332     if (fp)
333     {
```

```
        for (i = 0; i < DICT_SIZE; i++)
        {
            for (JITrav = D.JobInformationD[i]; JITrav != NULL; JITrav = JITrav->next)
            {
                fwrite(&JITrav->data, sizeof(Model_JobInformation), 1, fp);
            }
        }

        // fseek(fp, 0L, SEEK_END);
        // if (ftell(fp) == -1)
        // {
        //     count = 0;
        // }
        // else
        // {
        //     count = ftell(fp) / sizeof(Model_JobInformation);
        // }

        // if (count != D.count[3])
        // {
        //     return false;
        // }

        fclose(fp);
    }
    else
    {
        return false;
    }

    return true;
}

bool pushIssueSalaryD(Dictionary D)
{
    PSIS STrav;
    int i, count;
    FILE *fp;

    // Issue Salary
    fp = fopen("Emp_IssueSalary.bin", "wb");
    if (fp)
    {
        for (i = 0; i < DICT_SIZE; i++)
        {
            for (STrav = D.IssueSalaryD[i]; STrav != NULL; STrav = STrav->next)
            {
                fwrite(&STrav->data, sizeof(Model_IssueSalary), 1, fp);
            }
        }

        // fseek(fp, 0L, SEEK_END);
        // if (ftell(fp) == -1)
        // {
        //     count = 0;
        // }
        // else
        // {
        //     count = ftell(fp) / sizeof(Model_IssueSalary);
```

```
        // }

        // if (count != D.count[2])
        // {
        //     return false;
        // }

        fclose(fp);
    }
    else
    {
        return false;
    }

    return true;
}

bool pushBonusD(Dictionary D)
{
    PSB BTrav;
    int i, count;
    FILE *fp;

    // Bonus Information
    fp = fopen("Emp_Bonus.bin", "wb");
    if (fp)
    {
        for (i = 0; i < DICT_SIZE; i++)
        {
            for (BTrav = D.BonusD[i]; BTrav != NULL; BTrav = BTrav->next)
            {
                fwrite(&BTrav->data, sizeof(Model_Bonus), 1, fp);
            }
        }

        // fseek(fp, 0L, SEEK_END);
        // if (ftell(fp) == -1)
        // {
        //     count = 0;
        // }
        // else
        // {
        //     count = ftell(fp) / sizeof(Model_Bonus);
        // }

        // if (count != D.count[1])
        // {

        //     return false;
        // }

        fclose(fp);
    }
    else
    {
        return false;
    }

    return true;
}
```

```
453
454 bool pushAttendanceD(Dictionary D)
455 {
456     PSA ATrav;
457     int i, count;
458     FILE *fp;
459
460     // Attendance Information
461     fp = fopen("Emp_Attendance.bin", "wb");
462     if (fp)
463     {
464         for (i = 0; i < DICT_SIZE; i++)
465         {
466             for (ATrav = D.AttendanceD[i]; ATrav != NULL; ATrav = ATrav->next)
467             {
468                 fwrite(&ATrav->data, sizeof(Model_Attendance), 1, fp);
469             }
470         }
471
472         // fseek(fp, 0L, SEEK_END);
473         // if (ftell(fp) == -1)
474         // {
475         //     count = 0;
476         // }
477         // else
478         // {
479         //     count = ftell(fp) / sizeof(Model_Attendance);
480         // }
481
482         // if (count != D.count[0])
483         // {
484         //     return false;
485         // }
486
487         fclose(fp);
488     }
489     else
490     {
491         return false;
492     }
493
494     return true;
495 }
496
497 void insertDefault(Dictionary *D)
498 {
499     Model_User defaultUser = {
500         1,
501         "employeefn",
502         "employeeln",
503         FEMALE,
504         "03/19/22",
505         YES,
506         "1234",
507         "1234",
508         "employee@gmail.com",
509         "employeeaddress",
510         EMPLOYER};
511
512     Model_Bonus defaultBonus = {
```

```
        1,
        1,
        "employeebonus",
        100,
        "03/22"};

    Model_Attendance defaultAttendance = {
        1,
        1,
        0,
        0,
        0,
        0,
        "03/22",
    };

    Model_IssueSalary defaultIssueSalary = {
        1,
        1,
        1000,
        0,
        "03/22"};

    Model_JobInformation defaultJobInformation = {
        1,
        1,
        "employee",
        "employeelocation",
        "1234",
        "03/19/2022",
        "employeedepartment",
        "employee@gmail.com",
        2500,
        0};

    insertUser(D, defaultUser);
    insertAttendance(D, defaultAttendance);
    insertBonus(D, defaultBonus);
    insertJobInformation(D, defaultJobInformation);
    insertIssueSalary(D, defaultIssueSalary);
}

/*-------------------------------- End of Dictionary Controller ------------------
---------------*/

/*-------------------------------- Start of Attendance Controller ----------------
---------------*/

Model_Attendance *searchAttendance(Dictionary D, ID employeeID, char period[])
{
    PSA trav, temp;
    int hashVal = hash(employeeID);

    for (trav = D.AttendanceD[hashVal]; trav != NULL && (trav->data.employeeID !=
employeeID || strcmp(trav->data.period, period) != 0); trav = trav->next)
    {
        if (trav->data.employeeID == employeeID && strcmp(trav->data.period, period)
== 0)
            break;
    }
```

```
569
570     if (trav != NULL)
571     {
572         return &trav->data;
573     }
574     else
575     {
576         return NULL;
577     }
578 }
579
580 Model_Attendance createAttendance(Dictionary D, int empID, char period[])
581 {
582     Model_Attendance sa;
583
584     if (searchUser(D, empID))
585     {
586         char dType[20] = "Attendance";
587         sa.attendanceID = getNewID(dType, D);
588         sa.employeeID = empID;
589         strcpy(sa.period, period);
590         sa.present = 0;
591         sa.absent = 0;
592         sa.leave = 0;
593         sa.overtime = 0;
594     }
595     else
596     {
597         sa.attendanceID = -1;
598     }
599
600     return sa;
601 }
602
603 bool insertAttendance(Dictionary *D, Model_Attendance data)
604 {
605     PSA *trav;
606     int hashVal = hash(data.employeeID);
607
608     for (trav = &(D->AttendanceD[hashVal]); *trav != NULL && ((*trav)-
    >data.employeeID != data.employeeID || strcmp((*trav)->data.period, data.period) !=
    0); trav = &(*trav)->next)
609     {
610         if ((*trav)->data.employeeID == data.employeeID && strcmp((*trav)-
    >data.period, data.period) == 0)
611             break;
612     }
613
614     if (*trav == NULL)
615     {
616         *trav = (PSA)malloc(sizeof(Model_Attendance) + 1);
617         if (trav != NULL)
618         {
619             (*trav)->data = data;
620             (*trav)->next = NULL;
621             D->count[0]++;
622         }
623         return true;
624     }
625     else
```

```
626     {
627         return false;
628     }
629 }
630 
631 bool deleteAttendance(Dictionary *D, int empID, char period[])
632 {
633     PSA *trav, temp;
634     int hashVal = hash(empID);
635 
636     for (trav = &(D->AttendanceD[hashVal]); *trav != NULL && ((*trav)->data.employeeID != empID || strcmp((*trav)->data.period, period) != 0); trav = &(*trav)->next)
637     {
638         if ((*trav)->data.employeeID == empID && strcmp((*trav)->data.period, period) == 0)
639             break;
640     }
641 
642     if (*trav == NULL)
643     {
644         return false;
645     }
646     else
647     {
648         temp = *trav;
649         *trav = (*trav)->next;
650         free(temp);
651         D->count[0]--;
652         return true;
653     }
654 }
655 
656 void displayAttendance(Dictionary *D, ID employeeID, char period[])
657 {
658     Model_Attendance *emp = searchAttendance(*D, employeeID, period);
659     if (emp)
660     {
661         printf(" ____________________________________________________\n\n");
662 
663         printf(" EMPLOYEE #%d - %s         \t\t\n\n", employeeID, period);
664         printf(" Attendance ID:       \t\t%d  \n", emp->attendanceID);
665         printf(" Employee ID:           \t%d  \n", emp->employeeID);
666         printf(" Present:               \t%d  \n", emp->present);
667         printf(" Absent                 \t%d  \n", emp->absent);
668         printf(" Leave:                 \t%d  \n", emp->leave);
669         printf(" Overtime:              \t%d  \n", emp->overtime);
670         printf(" Period:                \t%s  \n", emp->period);
671         printf(" ____________________________________________________\n\n");
672     }
673     else
674     {
675         printf("\n ERROR: Employee #%d attendance for period %s does not exist.\n", employeeID, period);
676         printf(" ____________________________________________________\n\n");
677     }
678 }
679 
680 bool setPresent(int employeeID, Dictionary *D, char period[])
681 {
```

```
682     int presentNum;
683     Model_Attendance *sa = searchAttendance(*D, employeeID, period);
684     if (sa)
685     {
686         printf(" Enter No. of Present Days: ");
687         scanf("%d", &presentNum);
688         sa->present += presentNum;
689         return true;
690     }
691     else
692     {
693         printf("\n ERROR: Employee #%d attendance for period %s does not exist.\n",
    employeeID, period);
694         return false;
695     }
696 }
697
698 bool setLeave(int employeeID, Dictionary *D, char period[])
699 {
700     int leaveNum;
701     Model_Attendance *sa = searchAttendance(*D, employeeID, period);
702     if (sa)
703     {
704         printf(" Enter No. of Leave Days: ");
705         scanf("%d", &leaveNum);
706         sa->leave += leaveNum;
707         return true;
708     }
709     else
710     {
711         printf("\n ERROR: Employee #%d attendance for period %s does not exist.\n",
    employeeID, period);
712         return false;
713     }
714 }
715
716 bool setAbsent(int employeeID, Dictionary *D, char period[])
717 {
718     int absentNum;
719     Model_Attendance *sa = searchAttendance(*D, employeeID, period);
720     if (sa)
721     {
722         printf(" Enter No. of Absent Days: ");
723         scanf("%d", &absentNum);
724         sa->absent += absentNum;
725
726         return true;
727     }
728     else
729     {
730         printf("\n ERROR: Employee #%d attendance for period %s does not exist.\n",
    employeeID, period);
731         return false;
732     }
733 }
734
735 bool setOvertime(int employeeID, Dictionary *D, char period[])
736 {
737     int otHours;
738     Model_Attendance *sa = searchAttendance(*D, employeeID, period);
```

```
739     if (sa)
740     {
741         printf(" Enter No. of Overtime Hours: ");
742         scanf("%d", &otHours);
743         sa->overtime += otHours;
744 
745         return true;
746     }
747     else
748     {
749         printf("\n ERROR: Employee #%d attendance for period %s does not exist.\n",
    employeeID, period);
750         return false;
751     }
752 }
753 
754 void displayAllAttendance(Dictionary D, char period[])
755 {
756     PSA trav;
757     int i;
758 
759     printf(" %4s___%14s___%12s___%8s___%7s___%8s___%9s___%7s\n",
760            "____",
761            "______________",
762            "____________",
763            "________",
764            "_______",
765            "________",
766            "_________",
767            "_______");
768     printf(" \t\t\t\t\tATTENDANCE\n\n");
769     printf(" %-4s | %-14s | %-12s | %-8s | %-7s | %-8s | %-9s | %-7s\n",
770            "#",
771            "ATTENDANCE ID",
772            "EMPLOYEE ID",
773            "PRESENT",
774            "ABSENT",
775            "LEAVE",
776            "OVERTIME",
777            "PERIOD");
778     printf(" %-4s | %-14s | %-12s | %-8s | %-7s | %-8s | %-9s | %-7s\n",
779            "",
780            "",
781            "",
782            "",
783            "",
784            "",
785            "",
786            "");
787 
788     for (i = 0; i < DICT_SIZE; i++)
789     {
790         for (trav = D.AttendanceD[i]; trav != NULL; trav = trav->next)
791         {
792             if (strcmp(trav->data.period, period) == 0)
793             {
794                 printf(" %4d | %14d | %12d | %8d | %7d | %8d | %9d | %-7s\n",
795                        i,
796                        trav->data.attendanceID,
797                        trav->data.employeeID,
```

```
798                         trav->data.present,
799                         trav->data.absent,
800                         trav->data.leave,
801                         trav->data.overtime,
802                         trav->data.period);
803             }
804         }
805     }
806 
807     printf(" %4s_|_%14s_|_%12s_|_%8s_|_%7s_|_%8s_|_%9s_|_%7s\n",
808            "____",
809            "______________",
810            "____________",
811            "________",
812            "_______",
813            "________",
814            "_________",
815            "_______");
816     printf("\n End of Dictionary\n\n");
817 }
818 
819 void debugAttendance(Dictionary D)
820 {
821     PSA trav;
822     int i;
823     printf("\n ******************************************************\n");
824     printf(" (DEBUG) DICTIONARY ATTENDANCE\n");
825     printf(" ******************************************************\n");
826     printf(" %4s | %4s\n", "row", "ID");
827     for (i = 0; i < DICT_SIZE; i++)
828     {
829         printf(" %4d | ", i);
830         for (trav = D.AttendanceD[i]; trav != NULL; trav = trav->next)
831         {
832             printf(" ID#%d -> ", trav->data.attendanceID);
833         }
834         printf("\n", i);
835     }
836 }
837 
838 /*-------------------------------- End of Attendance Controller ------------------
    ---------------*/
839 
840 /*-------------------------------- Start of Bonus Controller ---------------------
    ---------------*/
841 
842 Model_Bonus createBonus(Dictionary D, int employeeID, char period[])
843 {
844     Model_Bonus bonus;
845 
846     if (searchUser(D, employeeID))
847     {
848         char dType[10] = "Bonus";
849         bonus.bonusID = getNewID(dType, D);
850         bonus.employeeID = employeeID;
851         strcpy(bonus.period, period);
852 
853         printf(" Bonus Name: ");
854         scanf("%s", &bonus.bonusName);
855         fflush(stdin);
```

```
856
857         printf(" Amount: ");
858         scanf("%lf", &bonus.amount);
859         fflush(stdin);
860     }
861     else
862     {
863         bonus.bonusID = -1;
864     }
865
866     return bonus;
867 }
868
869 bool insertBonus(Dictionary *D, Model_Bonus data)
870 {
871     PSB *trav;
872     int hashVal = hash(data.employeeID);
873
874     for (trav = &(D->BonusD[hashVal]); *trav != NULL && (data.employeeID != (*trav)->data.employeeID || strcmp((*trav)->data.period, data.period) != 0); trav = &(*trav)->next)
875     {
876         if (data.employeeID == (*trav)->data.employeeID && strcmp((*trav)->data.period, data.period) == 0)
877             break;
878     }
879
880     if (*trav == NULL)
881     {
882         *trav = (PSB)malloc(sizeof(Model_Bonus) + 1);
883         if (*trav != NULL)
884         {
885             (*trav)->data = data;
886             (*trav)->next = NULL;
887
888             D->count[1]++;
889         }
890         return true;
891     }
892     else
893     {
894         return false;
895     }
896 }
897
898 bool deleteBonus(Dictionary *D, ID employeeID, char period[])
899 {
900     PSB *trav, temp;
901     int hashVal = hash(employeeID);
902
903     for (trav = &(D->BonusD[hashVal]); *trav != NULL && ((*trav)->data.employeeID != employeeID || strcmp((*trav)->data.period, period) != 0); trav = &(*trav)->next)
904     {
905         if ((*trav)->data.employeeID == employeeID && strcmp((*trav)->data.period, period) == 0)
906             break;
907     }
908
909     if (*trav == NULL)
910     {
```

```
911         return false;
912     }
913     else
914     {
915         temp = *trav;
916         *trav = (*trav)->next;
917         free(temp);
918         D->count[1]--;
919         return true;
920     }
921 }
922 
923 Model_Bonus *searchBonus(Dictionary D, ID employeeID, char period[])
924 {
925     PSB trav, temp;
926     int hashVal = hash(employeeID);
927 
928     for (trav = D.BonusD[hashVal]; trav != NULL && (trav->data.employeeID != 
employeeID || strcmp(trav->data.period, period) != 0); trav = trav->next)
929     {
930         if (trav->data.employeeID == employeeID && strcmp(trav->data.period, period) 
== 0)
931             break;
932     }
933 
934     if (trav != NULL)
935     {
936         return &trav->data;
937     }
938     else
939     {
940         return NULL;
941     }
942 }
943 
944 void debugBonus(Dictionary D)
945 {
946     PSB trav;
947     int i;
948     printf("\n ******************************************************\n");
949     printf(" (DEBUG) DICTIONARY BONUS\n");
950     printf("\n ******************************************************\n");
951     printf(" %4s | %4s\n", "row", "ID");
952     for (i = 0; i < DICT_SIZE; i++)
953     {
954         printf(" %4d | ", i);
955         for (trav = D.BonusD[i]; trav != NULL; trav = trav->next)
956         {
957             printf(" ID#%d -> ", trav->data.bonusID);
958         }
959         printf("\n", i);
960     }
961 }
962 
963 void displayAllBonus(Dictionary D, char period[])
964 {
965     PSB trav;
966     int i;
967 
968     printf(" %-4s___%-9s___%-12s___%-20s___%-12s___%-7s\n",
```

```
969             "____",
970             "_________",
971             "____________",
972             "____________________",
973             "____________",
974             "_______");
975     printf("\t\t\t\tBONUS\n\n");
976     printf(" %-4s | %-9s | %-12s | %-20s | %-12s | %-7s\n",
977             "#",
978             "BONUS ID",
979             "EMPLOYEE ID",
980             "BONUS NAME",
981             "AMOUNT",
982             "PERIOD");
983     printf(" %-4s | %-9s | %-12s | %-20s | %-12s | %-7s\n",
984             "",
985             "",
986             "",
987             "",
988             "",
989             "");
990
991     for (i = 0; i < DICT_SIZE; i++)
992     {
993         for (trav = D.BonusD[i]; trav != NULL; trav = trav->next)
994         {
995             if (strcmp(trav->data.period, period) == 0)
996             {
997                 printf(" %-4d | %-9d | %-12d | %-20s | %-12.2lf | %-7s \n",
998                        i,
999                        trav->data.bonusID,
1000                       trav->data.employeeID,
1001                       trav->data.bonusName,
1002                       trav->data.amount,
1003                       trav->data.period);
1004            }
1005        }
1006    }
1007
1008    printf(" %-4s_|_%-9s_|_%-12s_|_%-20s_|_%-12s_|_%-7s\n",
1009            "____",
1010            "_________",
1011            "____________",
1012            "____________________",
1013            "____________",
1014            "_______");
1015    printf("\n End of Dictionary\n\n");
1016}
1017
1018void displayBonus(int employeeID, Dictionary *D, char period[])
1019{
1020    Model_Bonus *emp = searchBonus(*D, employeeID, period);
1021    if (emp)
1022    {
1023        printf(" ____________________________________________________________\n\n");
1024        printf(" EMPLOYEE #%d - %s         \t\t\n\n", employeeID, period);
1025
1026        printf(" Bonus ID:       \t%d", emp->bonusID);
1027        printf("\n Employee ID:    \t%d", emp->employeeID);
1028        printf("\n Bonus Name:     \t%s", emp->bonusName);
```

```
1029        printf("\n Amount          \t%.2lf", emp->amount);
1030        printf("\n Period:         \t%s\n", emp->period);
1031        printf(" _____________________________________________________\n\n");
1032    }
1033    else
1034    {
1035        printf("\n ERROR: Employee #%d bonus for period %s does not exist.\n",
    employeeID, period);
1036        printf(" _____________________________________________________\n\n");
1037    }
1038}
1039
1040/*---------------------------------- End of Bonus Controller ----------------------
    ---------------*/
1041
1042/*------------------------------ Start of Issue Salary Controller ----------------
    ---------------*/
1043
1044Model_IssueSalary createIssueSalary(Dictionary D, int employeeID, double balance,
    double pagibig, char period[])
1045{
1046    Model_IssueSalary is;
1047
1048    char dType[15] = "IssueSalary";
1049    is.issueID = getNewID(dType, D);
1050    is.employeeID = employeeID;
1051    is.balance = balance;
1052    is.pagibigBalance = pagibig;
1053    strcpy(is.period, period);
1054
1055    return is;
1056}
1057
1058bool insertIssueSalary(Dictionary *D, Model_IssueSalary data)
1059{
1060    PSIS *trav;
1061    int hashVal = hash(data.employeeID);
1062
1063    for (trav = &(D->IssueSalaryD[hashVal]); *trav != NULL && (data.employeeID !=
    (*trav)->data.employeeID || strcmp((*trav)->data.period, data.period) != 0); trav =
    &(*trav)->next)
1064    {
1065        if (data.employeeID == (*trav)->data.employeeID && strcmp((*trav)-
    >data.period, data.period) == 0)
1066            break;
1067    }
1068
1069    if (*trav == NULL)
1070    {
1071        *trav = (PSIS)malloc(sizeof(Model_IssueSalary) + 1);
1072        if (*trav != NULL)
1073        {
1074            (*trav)->data = data;
1075            (*trav)->next = NULL;
1076            D->count[2]++;
1077        }
1078        return true;
1079    }
1080    else
1081    {
```

```
1082        return false;
1083    }
1084}
1085
1086bool deleteIssueSalary(Dictionary *D, ID employeeID, char period[])
1087{
1088    PSIS *trav, temp;
1089    int hashVal = hash(employeeID);
1090
1091    for (trav = &(D->IssueSalaryD[hashVal]); *trav != NULL && ((*trav)->data.employeeID != employeeID || strcmp((*trav)->data.period, period) != 0); trav = &(*trav)->next)
1092    {
1093        if ((*trav)->data.employeeID == employeeID && strcmp((*trav)->data.period, period) == 0)
1094            break;
1095    }
1096
1097    if (*trav == NULL)
1098    {
1099        return false;
1100    }
1101    else
1102    {
1103        // Update pagibig deposit
1104        Model_JobInformation *ji = searchJobInformation(*D, employeeID);
1105        if (ji)
1106        {
1107            ji->pagibigDeposit -= (*trav)->data.pagibigBalance;
1108        }
1109        else
1110        {
1111            printf("\n Error: Failed to update pagibig deposit balance.\n");
1112        }
1113
1114        temp = *trav;
1115        *trav = (*trav)->next;
1116        free(temp);
1117        D->count[2]--;
1118        return true;
1119    }
1120}
1121
1122Model_IssueSalary *searchIssueSalary(Dictionary D, ID employeeID, char period[])
1123{
1124    PSIS trav, temp;
1125    int hashVal = hash(employeeID);
1126
1127    for (trav = D.IssueSalaryD[hashVal]; trav != NULL && (trav->data.employeeID != employeeID || strcmp(trav->data.period, period) != 0); trav = trav->next)
1128    {
1129        if (trav->data.employeeID == employeeID && strcmp(trav->data.period, period) == 0)
1130            break;
1131    }
1132
1133    if (trav != NULL)
1134    {
1135        return &trav->data;
1136    }
```

```
    else
    {
        return NULL;
    }
}

void debugSalary(Dictionary D)
{
    PSIS trav;
    int i;
    printf("\n ***************************************************\n");
    printf(" (DEBUG) DICTIONARY SALARY\n");
    printf("\n ***************************************************\n");
    printf(" %4s | %4s\n", "row", "ID");
    for (i = 0; i < DICT_SIZE; i++)
    {
        printf(" %4d | ", i);
        for (trav = D.IssueSalaryD[i]; trav != NULL; trav = trav->next)
        {
            printf(" ID#%d -> ", trav->data.issueID);
        }
        printf("\n", i);
    }
}

void displayAllSalary(Dictionary D, char period[])
{
    PSIS trav;
    int i;

    printf(" %-4s___%-9s___%-12s___%-8s___%8s___%-7s \n",
           "____",
           "_________",
           "____________",
           "________",
           "________",
           "_______");
    printf("\t\t\t SALARY\n\n");
    printf(" %-4s | %-9s | %-12s | %-8s | %-8s | %-7s \n",
           "#",
           "ISSUE ID",
           "EMPLOYEE ID",
           "BALANCE",
           "PAGIBIG",
           "PERIOD");
    printf(" %-4s | %-9s | %-12s | %-8s | %-8s | %-7s\n",
           "",
           "",
           "",
           "",
           "",
           "");
    for (i = 0; i < DICT_SIZE; i++)
    {
        for (trav = D.IssueSalaryD[i]; trav != NULL; trav = trav->next)
        {
            if (strcmp(trav->data.period, period) == 0)
            {
                printf(" %-4d | %-9d | %-12d | %-8.2lf | %-8.2lf | %-7s \n",
                       i,
```

```
                        trav->data.issueID,
                        trav->data.employeeID,
                        trav->data.balance,
                        trav->data.pagibigBalance,
                        trav->data.period);
            }
        }
    }

    printf(" %-4s_|_%-9s_|_%-12s_|_%-8s_|_%-8s_|_%-7s \n",
           "____",
           "_________",
           "____________",
           "________",
           "________",
           "_______");
    printf("\n End of Dictionary\n");
}

void displayIssueSalary(Dictionary *D, ID empID, char period[])
{
    Model_IssueSalary *emp = searchIssueSalary(*D, empID, period);
    if (emp)
    {
        printf(" ____________________________________________________\n\n");
        printf(" EMPLOYEE #%d - %s         \t\t\n\n", empID, period);
        printf(" Issue ID:          \t%d  \n", emp->issueID);
        printf(" Employee ID:       \t%d  \n", emp->employeeID);
        printf(" Balance:           \t%.2lf  \n", emp->balance);
        printf(" Pagibig:           \t%.2lf  \n", emp->pagibigBalance);
        printf(" Period:            \t%s  \n", emp->period);
        printf(" ____________________________________________________\n\n");
    }
    else
    {
        printf(" \n ERROR: Employee #%d salary for period %s does not exist.\n",
empID, period);
        printf(" ____________________________________________________\n\n");
    }
}

bool issueSalary(Dictionary *D, int empID, char period[])
{
    int temp;
    int present;
    int leave;
    int absent;
    int overtime;
    double additions = 0;
    double deductions = 0;
    double basicSalary;
    double dailyRate;
    double hourlyRate;
    double pagibigDeposit;
    double pagibig = 0;
    double income;
    double tax;

    Model_Attendance *sa = searchAttendance(*D, empID, period);
    if (sa)
```

```
1256    {
1257        present = sa->present;
1258        leave = sa->leave;
1259        absent = sa->absent;
1260        overtime = sa->overtime;
1261    }
1262    else
1263    {
1264        printf("\n ERROR: Employee #%d attendance for period %s does not exist.",
    empID, period);
1265        return false;
1266    }
1267
1268    Model_JobInformation *ji = searchJobInformation(*D, empID);
1269    if (ji)
1270    {
1271        basicSalary = ji->basicSalary;
1272        pagibigDeposit = ji->pagibigDeposit;
1273    }
1274    else
1275    {
1276        printf("\n ERROR: Employee #%d Job Information could not be found.", empID);
1277        return false;
1278    }
1279
1280    Model_Bonus *b = searchBonus(*D, empID, period);
1281    // check is employee has bonus
1282    if (b)
1283        additions += b->amount;
1284
1285    dailyRate = basicSalary / 30;
1286    hourlyRate = dailyRate / 8;
1287
1288    // check if employee has absences
1289    if (leave != absent)
1290        deductions += (dailyRate * absent);
1291
1292    // check if employee has overtime
1293    if (overtime > 0)
1294        additions += ((hourlyRate * 1.25) * overtime);
1295
1296    Model_IssueSalary *is = searchIssueSalary(*D, empID, period);
1297    if (!is)
1298    {
1299        income = ((basicSalary + additions) - deductions);
1300        printf(" ____________________________________________________\n\n");
1301        printf(" Basic Income: P%.2f\n", basicSalary);
1302        printf(" Taxable Income: P%.2f\n", income);
1303        if (b)
1304            printf(" Bonus: P%.2f\n", b->amount);
1305        printf(" Additions: P%.2f\n", additions);
1306        printf(" Deductions: P%.2f\n", deductions);
1307
1308        tax = calculateTax(basicSalary, income, &pagibigDeposit, &pagibig);
1309        income -= (tax * -1);
1310
1311        printf(" Tax: P%.2f\n", tax);
1312        printf(" Issue Salary: P%.2f\n", income);
1313        printf(" ____________________________________________________\n\n");
```

```
1314        printf(" Add Issue Salary to the employee's record for period %s?\n\n",
    period);
1315        printf(" [1] Yes\n");
1316        printf(" [2] No\n");
1317        printf(" ____________________________________________________\n\n");
1318        printf(" Select Option: ");
1319        scanf("%d", &temp);
1320
1321        if (temp)
1322        {
1323            // Create new issue salary
1324            Model_IssueSalary is2 = createIssueSalary(*D, empID, income, pagibig,
    period);
1325            insertIssueSalary(D, is2);
1326
1327            // Update pagibig deposit
1328            Model_JobInformation *ji2 = searchJobInformation(*D, empID);
1329            if (ji2)
1330            {
1331                ji2->pagibigDeposit = pagibigDeposit;
1332            }
1333            else
1334            {
1335                printf("\n Error: Failed to update pagibig deposit.\n");
1336            }
1337            return true;
1338        }
1339        else if (temp == 2)
1340        {
1341            return false;
1342        }
1343        else
1344        {
1345            printf("\n ERROR: Input not recognized!");
1346            return false;
1347        }
1348    }
1349    else
1350    {
1351        printf("\n ERROR: Issue Salary for period %s already exists!", period);
1352        return false;
1353    }
1354 }
1355
1356 double calculateTax(double basicIncome, double taxableIncome, double
    *pagibigDeposit, double *pagibig)
1357 {
1358     double tax = 0;
1359
1360     // SSS (4%)
1361     tax += taxableIncome * 0.04;
1362
1363     if (debug)
1364     {
1365         printf("\n (Debug) Tax + SSS = %.2lf", tax);
1366     }
1367
1368     // Pag-ibig (1% or 2%, max P24,300 per year)
1369     if (*pagibigDeposit < 24300)
1370     {
```

```
1371        // Calculate the Pagibig tax for current month
1372        double pagibigTax = taxableIncome < 1500 ? taxableIncome * 0.01 :
    taxableIncome * 0.02;
1373
1374        if (debug)
1375        {
1376            printf("\n (Debug) Pagibig Tax = %.2lf", pagibigTax);
1377        }
1378
1379        // Add Pagibig tax such that Pagibig deposit doesn't exceed 24300
1380        if (*pagibigDeposit + pagibigTax > 24300)
1381        {
1382            tax += 24300 - *pagibigDeposit;
1383            *pagibigDeposit += 24300 - *pagibigDeposit;
1384            *pagibig = 24300 - *pagibigDeposit;
1385        }
1386        else
1387        {
1388            tax += pagibigTax;
1389            *pagibigDeposit += pagibigTax;
1390            *pagibig = pagibigTax;
1391        }
1392
1393        if (debug)
1394        {
1395            printf("\n (Debug) Tax + Pagibig = %.2lf", tax);
1396        }
1397    }
1398
1399    // PHIC (1.75%)
1400    tax += taxableIncome * 0.0175;
1401
1402    if (debug)
1403    {
1404        printf("\n (Debug) Tax + PHIC = %.2lf", tax);
1405    }
1406
1407    // WISP (P225)
1408    tax -= 225;
1409
1410    // Calculate annual tax based on monthly income
1411    double annualTax = 0;
1412    double annualSalary = basicIncome * 12;
1413
1414    if (debug)
1415    {
1416        printf("\n (Debug) Annual Salary = %.2lf", annualSalary);
1417    }
1418
1419    if (annualSalary <= 250000)
1420    {
1421        // 0%
1422        annualTax += (annualSalary * 0);
1423    }
1424    else if (annualSalary <= 400000)
1425    {
1426        // 20%
1427        annualTax += (annualSalary * 0.2);
1428    }
1429    else if (annualSalary <= 800000)
```

```
    {
        // P30,000 + 25%
        annualTax += 30000 + ((annualSalary - 30000) * 0.25);
    }
    else if (annualSalary <= 2000000)
    {
        // P130,000 + 30%
        annualTax += 130000 + ((annualSalary - 130000) * 0.30);
    }
    else if (annualSalary <= 8000000)
    {
        // P490,000 + 32%
        annualTax += 490000 + ((annualSalary - 490000) * 0.32);
    }
    else if (annualSalary > 8000000)
    {
        // P2,410,000 + 35%
        annualTax += 2410000 + ((annualSalary - 2410000) * 0.35);
    }

    // Convert annual tax to monthly
    tax += (annualTax / 12);

    if (debug)
    {
        printf("\n (Debug) Annual Tax = %.2lf", annualTax);
        printf("\n (Debug) Monthly Tax = %.2lf", annualTax / 12);
    }

    return tax;
}

/*------------------------------- End of Issue Salary Controller ------------------
---------------*/

/*----------------------------- Start of Job Information Controller ---------------
---------------*/

Model_JobInformation createJobInformation(Dictionary D, ID employeeID)
{
    Model_JobInformation jobInfo;

    printf("\n CREATE EMPLOYEE JOB INFORMATION\n\n");

    char dType[15] = "JobInformation";
    jobInfo.employmentID = getNewID(dType, D);
    jobInfo.employeeID = employeeID;

    printf(" Job Position: ");
    scanf("%s", &jobInfo.jobPosition);
    fflush(stdin);

    printf(" Job Location: ");
    scanf("%s", &jobInfo.jobLocation);
    fflush(stdin);

    printf(" Job Phone (11 digits): ");
    scanf("%s", &jobInfo.jobPhone);
    fflush(stdin);

```

```
1488      printf(" Start Date (mm/dd/yy): ");
1489      scanf("%s", &jobInfo.startDate);
1490      fflush(stdin);
1491
1492      printf(" Department: ");
1493      scanf("%s", &jobInfo.department);
1494      fflush(stdin);
1495
1496      printf(" Job Email: ");
1497      scanf("%s", &jobInfo.jobEmail);
1498      fflush(stdin);
1499
1500      printf(" Basic Salary: ");
1501      scanf("%lf", &jobInfo.basicSalary);
1502      fflush(stdin);
1503
1504      printf(" Pagibig Deposit: ");
1505      scanf("%lf", &jobInfo.pagibigDeposit);
1506      fflush(stdin);
1507
1508      return jobInfo;
1509  }
1510
1511  bool insertJobInformation(Dictionary *D, Model_JobInformation data)
1512  {
1513      PSJI *trav;
1514      int hashVal = hash(data.employeeID);
1515
1516      for (trav = &(D->JobInformationD[hashVal]); *trav != NULL && (*trav)-
      >data.employmentID != data.employmentID; trav = &(*trav)->next)
1517      {
1518      }
1519
1520      if (*trav == NULL)
1521      {
1522          *trav = (PSJI)malloc(sizeof(Model_JobInformation) + 1);
1523          if (*trav != NULL)
1524          {
1525              (*trav)->data = data;
1526              (*trav)->next = NULL;
1527              D->count[3]++;
1528          }
1529          return true;
1530      }
1531      else
1532      {
1533          return false;
1534      }
1535  }
1536
1537  bool deleteJobInformation(Dictionary *D, ID employeeID)
1538  {
1539      PSJI *trav, temp;
1540      int hashVal = hash(employeeID);
1541
1542      for (trav = &(D->JobInformationD[hashVal]); *trav != NULL && (*trav)-
      >data.employeeID != employeeID; trav = &(*trav)->next)
1543      {
1544      }
1545
```

```
1546    if (*trav == NULL)
1547    {
1548        return false;
1549    }
1550    else
1551    {
1552        temp = *trav;
1553        *trav = (*trav)->next;
1554        free(temp);
1555        D->count[3]--;
1556        return true;
1557    }
1558 }
1559
1560 Model_JobInformation *searchJobInformation(Dictionary D, ID employeeID)
1561 {
1562    PSJI trav, temp;
1563    int hashVal = hash(employeeID);
1564
1565    for (trav = D.JobInformationD[hashVal]; trav != NULL && trav->data.employeeID !=
     employeeID; trav = trav->next)
1566    {
1567    }
1568
1569    if (trav != NULL)
1570    {
1571        return &trav->data;
1572    }
1573    else
1574    {
1575        return NULL;
1576    }
1577 }
1578
1579 void debugJobInformation(Dictionary D)
1580 {
1581    PSJI trav;
1582    int i;
1583    printf("\n ******************************************************\n");
1584    printf(" (DEBUG) DICTIONARY JOB INFORMATION\n");
1585    printf("\n ******************************************************\n");
1586    printf(" %4s | %4s\n", "row", "ID");
1587    for (i = 0; i < DICT_SIZE; i++)
1588    {
1589        printf(" %4d | ", i);
1590        for (trav = D.JobInformationD[i]; trav != NULL; trav = trav->next)
1591        {
1592            printf(" ID#%d -> ", trav->data.employmentID);
1593        }
1594        printf("\n", i);
1595    }
1596 }
1597
1598 void displayAllJobInformation(Dictionary D)
1599 {
1600    PSJI trav;
1601    int i;
1602
1603    printf("
     %-4s___%-14s___%-14s___%-13s___%-20s___%-20s___%-20s___%-20s___%-13s___%-16s \n",
```

```
1604            "____",
1605            "_______________",
1606            "_______________",
1607            "_______________",
1608            "____________________",
1609            "____________________",
1610            "____________________",
1611            "____________________",
1612            "_____________",
1613            "_________________");
1614        printf("
                JOB INFORMATION\n\n");
1615        printf(" %-4s | %-14s | %-14s | %-13s | %-20s | %-20s | %-20s | %-20s | %-13s |
    %-13s \n",
1616            "#",
1617            "EMPLOYMENT ID",
1618            "EMPLOYEE ID",
1619            "JOB POSITION",
1620            "JOB LOCATION",
1621            "JOB PHONE",
1622            "DEPARTMENT",
1623            "JOB EMAIL",
1624            "BASIC SALARY",
1625            "PAGIBIG DEPOSIT");
1626        printf(" %-4s | %-14s | %-14s | %-13s | %-20s | %-20s | %-20s | %-20s | %-13s |
    %-13s \n",
1627            "",
1628            "",
1629            "",
1630            "",
1631            "",
1632            "",
1633            "",
1634            "",
1635            "",
1636            "");
1637
1638        for (i = 0; i < DICT_SIZE; i++)
1639        {
1640            for (trav = D.JobInformationD[i]; trav != NULL; trav = trav->next)
1641            {
1642                printf(" %-4d | %-14d | %-14d | %-13s | %-20s | %-20s | %-20s | %-20s |
    %-13.2lf | %-13.2lf \n",
1643                       i,
1644                       trav->data.employmentID,
1645                       trav->data.employeeID,
1646                       trav->data.jobPosition,
1647                       trav->data.jobLocation,
1648                       trav->data.jobPhone,
1649                       trav->data.department,
1650                       trav->data.jobEmail,
1651                       trav->data.basicSalary,
1652                       trav->data.pagibigDeposit);
1653            }
1654        }
1655
1656        printf("
    %-4s_|_%-14s_|_%-14s_|_%-13s_|_%-20s_|_%-20s_|_%-20s_|_%-20s_|_%-13s_|_%-16s \n",
1657            "____",
1658            "_______________",
```

```
1659            "_____________",
1660            "_____________",
1661            "____________________",
1662            "____________________",
1663            "____________________",
1664            "____________________",
1665            "_____________",
1666            "_________________");
1667
1668    printf("\n End of Dictionary\n\n");
1669}
1670
1671void displayJobInformation(ID employeeID, Dictionary *D)
1672{
1673    Model_JobInformation *ji = searchJobInformation(*D, employeeID);
1674    if (ji)
1675    {
1676        printf(" ____________________________________________________\n\n");
1677        printf(" EMPLOYEE #%d         \t\t\n\n", employeeID);
1678
1679        printf(" Employee ID:      \t%d   \n", ji->employeeID);
1680        printf(" Employment ID:    \t%d   \n", ji->employmentID);
1681        printf(" Job Position:     \t%s   \n", ji->jobPosition);
1682        printf(" Job Location:     \t%s   \n", ji->jobLocation);
1683        printf(" Job Phone:        \t%s   \n", ji->jobPhone);
1684        printf(" Job Email:        \t%s   \n", ji->jobEmail);
1685        printf(" Start Date:       \t%s   \n", ji->startDate);
1686        printf(" Department:       \t%s   \n", ji->department);
1687        printf(" Basic Salary:     \t%.2lf  \n", ji->basicSalary);
1688        printf(" Pag-ibig Deposit: \t%.2lf  \n", ji->pagibigDeposit);
1689        printf(" ____________________________________________________\n");
1690    }
1691    else
1692    {
1693        printf("\n ERROR: Employee ID %d not found.\n", employeeID);
1694        printf(" ____________________________________________________\n");
1695    }
1696}
1697
1698/*------------------------------ End of Job Information Controller ----------------
    ---------------*/
1699
1700/*--------------------------------- Start of User Controller ---------------------
    ---------------*/
1701
1702Model_User createUserInformation(Dictionary D)
1703{
1704    Model_User emp;
1705
1706    char dType[10] = "User";
1707    emp.userID = getNewID(dType, D);
1708
1709    printf(" First Name: ");
1710    scanf("%s", &emp.firstName);
1711    fflush(stdin);
1712
1713    printf(" Last Name: ");
1714    scanf("%s", &emp.lastName);
1715    fflush(stdin);
1716
```

```
1717    printf(" Gender [MALE(0) / FEMALE(1)]: ");
1718    scanf("%u", &emp.gender);
1719    fflush(stdin);
1720
1721    printf(" Date of Birth (mm/dd/yy): ");
1722    scanf("%s", &emp.dateOfBirth);
1723    fflush(stdin);
1724
1725    printf(" Filipino Citizen [NO(0) / YES(1)]: ");
1726    scanf("%u", &emp.filipinoCitizen);
1727    fflush(stdin);
1728
1729    printf(" Home Phone (7 digits): ");
1730    scanf("%s", &emp.homePhone);
1731    fflush(stdin);
1732
1733    printf(" Mobile Phone (11 digits): ");
1734    scanf("%s", &emp.mobilePhone);
1735    fflush(stdin);
1736
1737    printf(" Email Address: ");
1738    scanf("%s", &emp.emailAddress);
1739    fflush(stdin);
1740
1741    printf(" Address: ");
1742    scanf("%s", &emp.address);
1743    fflush(stdin);
1744
1745    printf(" User Type [EMPLOYEE(0) / EMPLOYER(1)]: ");
1746    scanf("%u", &emp.userType);
1747    fflush(stdin);
1748
1749    return emp;
1750 }
1751
1752 bool insertUser(Dictionary *D, Model_User data)
1753 {
1754     PSU *trav;
1755     int hashVal = hash(data.userID);
1756
1757     for (trav = &D->UserD[hashVal]; *trav != NULL && strcmp((*trav)-
    >data.emailAddress, data.emailAddress) != 0; trav = &(*trav)->next)
1758     {
1759     }
1760
1761     if (*trav == NULL)
1762     {
1763         *trav = (PSU)malloc(sizeof(Model_User) + 1);
1764         if (*trav != NULL)
1765         {
1766             (*trav)->data = data;
1767             (*trav)->next = NULL;
1768
1769             D->count[4]++;
1770         }
1771         return true;
1772     }
1773     else
1774     {
1775         return false;
```

```
1776        }
1777    }
1778
1779    bool deleteUser(Dictionary *D, ID userID)
1780    {
1781        PSU *trav, temp;
1782        int hashVal = hash(userID);
1783
1784        for (trav = &(D->UserD[hashVal]); *trav != NULL && (*trav)->data.userID !=
        userID; trav = &(*trav)->next)
1785        {
1786        }
1787
1788        if (*trav == NULL)
1789        {
1790            return false;
1791        }
1792        else
1793        {
1794            temp = *trav;
1795            *trav = (*trav)->next;
1796            free(temp);
1797            D->count[4]--;
1798            return true;
1799        }
1800    }
1801
1802    Model_User *searchUser(Dictionary D, ID userID)
1803    {
1804        PSU trav, temp;
1805        int hashVal = hash(userID);
1806
1807        for (trav = D.UserD[hashVal]; trav != NULL && trav->data.userID != userID; trav
        = trav->next)
1808        {
1809        }
1810
1811        if (trav != NULL)
1812        {
1813            return &trav->data;
1814        }
1815        else
1816        {
1817            return NULL;
1818        }
1819    }
1820
1821    void debugUser(Dictionary D)
1822    {
1823        PSU trav;
1824        int i;
1825        printf("\n ***************************************************\n");
1826        printf(" (DEBUG) DICTIONARY USER\n");
1827        printf("\n ***************************************************\n");
1828        printf(" %4s | %4s\n", "row", "ID");
1829        for (i = 0; i < DICT_SIZE; i++)
1830        {
1831            printf(" %4d | ", i);
1832            for (trav = D.UserD[i]; trav != NULL; trav = trav->next)
1833            {
```

```
1834            printf(" ID#%d -> ", trav->data.userID);
1835        }
1836        printf("\n", i);
1837    }
1838}
1839
1840void displayAllUser(Dictionary D)
1841{
1842    PSU trav;
1843    int i;
1844
1845    printf("
%-4s___%-8s___%-11s___%-10s___%-7s___%-14s___%-17s___%-10s___%-13s___%-20s___%-20s__
_%-10s \n",
1846           "____",
1847           "________",
1848           "___________",
1849           "__________",
1850           "_______",
1851           "______________",
1852           "__________________",
1853           "__________",
1854           "______________",
1855           "____________________",
1856           "____________________",
1857           "__________");
1858    printf("\t\t\t\t\t\t\t\t\t\t\tUSER\n\n");
1859    printf(" %-4s | %-8s | %-11s | %-10s | %-7s | %-14s | %-17s | %-10s | %-13s |
%-20s | %-20s | %-10s \n",
1860           "#",
1861           "USER ID",
1862           "FIRST NAME",
1863           "LAST NAME",
1864           "GENDER",
1865           "DATE OF BIRTH",
1866           "FILIPINO CITIZEN",
1867           "HOME PAGE",
1868           "MOBILE PHONE",
1869           "EMAIL ADDRESS",
1870           "ADDRESS",
1871           "USER TYPE");
1872    printf(" %-4s | %-8s | %-11s | %-10s | %-7s | %-14s | %-17s | %-10s | %-13s |
%-20s | %-20s | %-10s \n",
1873           "",
1874           "",
1875           "",
1876           "",
1877           "",
1878           "",
1879           "",
1880           "",
1881           "",
1882           "",
1883           "",
1884           "");
1885
1886    for (i = 0; i < DICT_SIZE; i++)
1887    {
1888        for (trav = D.UserD[i]; trav != NULL; trav = trav->next)
1889        {
```

```
1890            char filipinoCitizen[10];
1891            char gender[10];
1892            char userType[10];
1893
1894            if (trav->data.gender == MALE)
1895            {
1896                strcpy(gender, "Male");
1897            }
1898            else
1899            {
1900                strcpy(gender, "Female");
1901            }
1902
1903            if (trav->data.filipinoCitizen == NO)
1904            {
1905
1906                strcpy(filipinoCitizen, "No");
1907            }
1908            else
1909            {
1910                strcpy(filipinoCitizen, "Yes");
1911            }
1912
1913            if (trav->data.userType == EMPLOYEE)
1914            {
1915
1916                strcpy(userType, "Employee");
1917            }
1918            else
1919            {
1920                strcpy(userType, "Employer");
1921            }
1922
1923            printf(" %-4d | %-8d | %-11s | %-10s | %-7s | %-14s | %-17s | %-10s |
    %-13s | %-20s | %-20s | %-10s \n",
1924                   i,
1925                   trav->data.userID,
1926                   trav->data.firstName,
1927                   trav->data.lastName,
1928                   gender,
1929                   trav->data.dateOfBirth,
1930                   filipinoCitizen,
1931                   trav->data.homePhone,
1932                   trav->data.mobilePhone,
1933                   trav->data.emailAddress,
1934                   trav->data.address,
1935                   userType);
1936        }
1937    }
1938
1939    printf("
    %-4s|_%-8s|_%-11s|_%-10s|_%-7s|_%-14s|_%-17s|_%-10s|_%-13s|_%-20s|_%-20s|
    _%-10s \n",
1940            "____",
1941            "________",
1942            "___________",
1943            "__________",
1944            "_______",
1945            "_____________",
1946            "________________",
```

```
1947            "__________",
1948            "______________",
1949            "____________________",
1950            "____________________",
1951            "__________");
1952    printf("\n End of Dictionary\n\n");
1953 }
1954
1955 void displayUserInformation(ID userID, Dictionary *D)
1956 {
1957     Model_User *emp = searchUser(*D, userID);
1958     if (emp)
1959     {
1960         printf(" _____________________________________________________\n\n");
1961         printf(" EMPLOYEE #%d        \t\t\n\n", userID);
1962         printf(" Employee ID:      \t%d  \n", emp->userID);
1963         printf(" First Name:       \t%s  \n", emp->firstName);
1964         printf(" Last Name:        \t%s  \n", emp->lastName);
1965
1966         if (emp->gender == MALE)
1967         {
1968             printf(" Gender:           \t%s  \n", "MALE");
1969         }
1970         else
1971         {
1972             printf(" Gender:           \t%s  \n", "FEMALE");
1973         }
1974
1975         printf(" Date of Birth     \t%s  \n", emp->dateOfBirth);
1976
1977         if (emp->filipinoCitizen == NO)
1978         {
1979             printf(" Filipino:           \t%s  \n", "NO");
1980         }
1981         else
1982         {
1983             printf(" Filipino:           \t%s  \n", "YES");
1984         }
1985
1986         printf(" Home Phone:       \t%s  \n", emp->homePhone);
1987         printf(" Mobile Phone:     \t%s  \n", emp->mobilePhone);
1988         printf(" Email:            \t%s  \n", emp->emailAddress);
1989         printf(" Address           \t%s  \n", emp->address);
1990         if (emp->userType == EMPLOYEE)
1991         {
1992             printf(" User Type:          \t%s  \n", "EMPLOYEE");
1993         }
1994         else
1995         {
1996             printf(" User Type:          \t%s  \n", "EMPLOYER");
1997         }
1998         printf(" _____________________________________________________\n\n");
1999     }
2000     else
2001     {
2002         printf("\n ERROR: Employee ID %d not found.", userID);
2003         printf("\n _____________________________________________________\n\n");
2004     }
2005 }
2006
```

```
2007 /*---------------------------------- End of User Controller ----------------------
     ---------------*/
2008
2009 /*---------------------------------- Start of Debug Controller -------------------
     ----------------*/
2010
2011 void displayDictionariesCount(Dictionary D)
2012 {
2013     printf(" ******************************************************\n");
2014     printf(" Dictionaries Count\n");
2015     printf(" ******************************************************\n");
2016     printf(" Attendance: %d\n", D.count[0]);
2017     printf(" Bonus: %d\n", D.count[1]);
2018     printf(" Issue Salary: %d\n", D.count[2]);
2019     printf(" Job Information: %d\n", D.count[3]);
2020     printf(" User: %d\n", D.count[4]);
2021     printf(" ******************************************************\n");
2022 }
2023
2024 /*---------------------------------- End of Debug Controller ----------------------
     ---------------*/
2025
2026 /*---------------------------------- Start of UI Controller ----------------------
     ---------------*/
2027
2028 void header(void)
2029 {
2030     printf(" _____________________________________________________________\n");
2031     printf("   _             _   _             __      __ ___ _       \n");
2032     printf("  |_) /\\ \\_/ |_) / \\ |  |    (_ \\_/ (_   | |_ |\\/| \n");
2033     printf("  |  /--\\ |  | \\ \\_/ |_ |_   __) |  __)  | |_ |  | \n");
2034     printf(" _____________________________________________________________\n\n");
2035 }
2036
2037 void invalidInput(void)
2038 {
2039     printf("\n ERROR: Please enter a vaid input.\n");
2040     printf(" _____________________________________________________________\n\n");
2041     printf(" *Press any key to continue...* ");
2042     getch();
2043 }
2044
2045 /*---------------------------------- End of UI Controller ----------------------
     ---------------*/
2046
2047 /*---------------------------------- End of Functions --------------------------
     ---------------*/
2048
```