**typedef struct Employee**

**pkert data type tmei named “Employee” hz typedef ng prer somrab do name “struct Employee” to just “Employee”.**

typedef struct Employee

{

int id;

char name[20];

char phone\_num[20];

char dob[20];

float salary;

int status;

} Employee;

**typedef struct NodeType**

**NodeType prer somrab dak Employee ng next (next derm3 point tv node muy tt)**

typedef struct NodeType

{

Employee e;

struct NodeType \*next;

} NodeType;

**NodeType \*initializeList();**

**Function ng prer somrab pkert linked list.**

NodeType \*initializeList()

{

NodeType \*origin = getNode();

origin->e.id = 0;

strcpy(origin->e.name, "GOD");

strcpy(origin->e.phone\_num, "Not Telling");

strcpy(origin->e.dob, "Before Universe");

origin->e.salary = 0;

origin->e.status = 0;

origin->next = NULL;

NodeType \*temp = origin;

// Get all employees from file

Employee \*emp = readFromFile();

// printf("%i\n", sizeof(NodeType));

int n = getEmployeesAmount();

for (int i = 0; i < n; i++)

{

// Handle last node of the linked list

if (i == n - 1)

{

temp->e = \*(emp + i);

temp->next = NULL;

break;

} else

{

temp->next = getNode();

temp->e = \*(emp + i);

temp = temp->next;

}

}

return origin;

}

**NodeType \*insertNode(NodeType \*origin);**

**Function ng somrab ptheam (insert) node muy nv kleng na muy (first in list, random position, and end of list)**

NodeType \*insertNode(NodeType \*origin)

{

int option = options("insert");

Employee e;

NodeType \*temp;

NodeType \*temp2 = origin;

switch (option)

{

case 0:

break;

case 1:

e = input();

temp = getNode();

temp->next = origin;

temp->e = e;

origin = temp;

break;

case 2:

int id;

printf("Employee ID: ");

fflush(stdin);

scanf("%i", &id);

while (origin != NULL)

{

if (origin->e.id == id)

{

temp->next = getNode();

temp->next->next = origin;

temp->next->e = input();

break;

}

temp = origin;

origin = origin->next;

}

origin = temp2;

break;

case 3:

temp = origin;

while (origin->next != NULL)

{

origin = origin->next;

}

origin->next = getNode();

origin->next->e = input();

origin->next->next = NULL;

origin = temp2;

break;

}

return origin;

}

**NodeType \*deleteNode(NodeType \*origin);**

**Function ng prer somrab delete note nv kleng muy (first node, random position, or last node)**

NodeType \*deleteNode(NodeType \*origin)

{

int option = options("delete");

NodeType \*temp;

switch (option)

{

case 1:

origin = origin->next;

break;

case 2:

int id;

printf("Employee ID: ");

fflush(stdin);

scanf("%i", &id);

while (origin != NULL)

{

if (origin->e.id == id)

{

temp->next = origin->next;

break;

}

temp = origin;

origin = origin->next;

}

origin = temp;

break;

case 3:

temp = origin;

while (origin->next->next != NULL)

{

origin = origin->next;

}

origin->next = NULL;

origin = temp;

break;

case 0:

break;

}

printf("Node has been deleted!\n");

return origin;

}

**void modifyNode(NodeType \*origin);**

**arch hav vea muy tt tha updateNode kor ban.**

**Prer vea derm3 do data pos employee na muy tam id, name, phone number, date of birth, salary, or status**

void modifyNode(NodeType \*origin)

{

int option = options("modify");

switch (option)

{

case 1:

int emp\_id;

printf("What ID is the employee: ");

fflush(stdin);

scanf("%i", &emp\_id);

while (origin != NULL)

{

if (origin->e.id == emp\_id)

{

switch (options("modify-field"))

{

case 1:

int id;

fflush(stdin);

printf("Enter new ID: ");

scanf("%i", &id);

origin->e.id = id;

break;

case 2:

char name[20];

printf("Enter new Name: ");

fflush(stdin);

gets(origin->e.name);

break;

case 3:

char p\_num[20];

printf("Enter new Phone Number: ");

fflush(stdin);

gets(origin->e.phone\_num);

break;

case 4:

char dob[20];

printf("Enter new Date of Birth: ");

fflush(stdin);

gets(origin->e.dob);

break;

case 5:

float sa;

printf("Enter new Salary: ");

fflush(stdin);

scanf("%f", &sa);

origin->e.salary = sa;

break;

case 6:

int s;

printf("Enter new Status: ");

fflush(stdin);

scanf("%s", &s);

origin->e.status = s;

break;

case 0:

break;

}

}

origin = origin->next;

}

break;

}

}

**void search(NodeType \*origin);**

**self-explanatory. Function search is used for searching an employee by id, name, phone number, date of birth, salary, or status**

void search(NodeType \*origin)

{

int option = options("search");

int emp\_id;

char name[20];

char p\_num[20];

char dob[20];

float salary;

int status;

switch (option)

{

case 1:

int opt = options("search-field");

switch (opt)

{

case 1:

printf("Employee ID: ");

fflush(stdin);

scanf("%i", &emp\_id);

while (origin != NULL)

{

if (origin->e.id == emp\_id)

{

printf("Found!\n");

return;

}

origin = origin->next;

}

printf("Not found!\n");

break;

case 2:

printf("Employee Name: ");

fflush(stdin);

scanf("%s", &name);

while (origin != NULL)

{

if (strcmpi(origin->e.name, name) == 0)

{

printf("Found!\n");

return;

}

origin = origin->next;

}

printf("Not found!\n");

break;

case 3:

printf("Employee Phone Number: ");

fflush(stdin);

scanf("%s", &p\_num);

while (origin != NULL)

{

if (strcmpi(origin->e.phone\_num, p\_num) == 0)

{

printf("Found!\n");

return;

}

origin = origin->next;

}

printf("Not found!\n");

break;

case 4:

printf("Employee Date of Birth: ");

fflush(stdin);

scanf("%s", &dob);

while (origin != NULL)

{

if (strcmpi(origin->e.dob, dob) == 0)

{

printf("Found!\n");

return;

}

origin = origin->next;

}

printf("Not found!\n");

break;

case 5:

printf("Employee Salary: ");

fflush(stdin);

scanf("%i", &salary);

while (origin != NULL)

{

if (origin->e.salary == salary)

{

printf("Found!\n");

return;

}

origin = origin->next;

}

printf("Not found!\n");

break;

case 6:

printf("Employee Status: ");

fflush(stdin);

scanf("%i", &status);

while (origin != NULL)

{

if (origin->e.status == status)

{

printf("Found!\n");

return;

}

origin = origin->next;

}

printf("Not found!\n");

break;

case 0:

break;

}

case 0:

break;

}

}

**int getEmployeesAmount();**

**aka. Count();**

**this function is used to get number of employee registered.**

int getEmployeesAmount()

{

FILE \*f = fopen("employee.bin", "rb");

// Get the byte size of the file

int n;

fseek(f, 0L, SEEK\_END);

n = ftell(f);

rewind(f);

// Get the amount of objects

n = n / sizeof(Employee);

fclose(f);

return n;

}