

## Print of Dynamic Array hash functions

```
1 Name : Fawzy
Age : 45
Salary : 5000
Experience in years : 8

2 Name : Mina
Age : 30
Salary : 10000
Experience in years : 4

3 Name : Yara
Age : 19
Salary : 2000
Experience in years : 0

4 Name : Mariam
Age : 32
Salary : 8000
Experience in years : 2

5 Name : Ayman
Age : 33
Salary : 4000
Experience in years : 8

6 Name : Aya
Age : 26
Salary : 6000
Experience in years : 3

7 Name : Abdallah
Age : 29
Salary : 7000
Experience in years : 4

8 Name : Fatma
Age : 21
Salary : 3000
Experience in years : 1

9 Name : Roshdy
Age : 28
Salary : 9000
Experience in years : 3
```

## Print of Linked List hash functions

```
-----  
1 Name : Fawzy  
Age : 45  
Salary : 5000  
Experience in years : 8  
  
2 Name : Yara  
Age : 19  
Salary : 2000  
Experience in years : 0  
  
3 Name : Ayman  
Age : 33  
Salary : 4000  
Experience in years : 8  
  
4 Name : Aya  
Age : 26  
Salary : 6000  
Experience in years : 3  
  
5 Name : Mina  
Age : 30  
Salary : 10000  
Experience in years : 4  
  
6 Name : Mariam  
Age : 32  
Salary : 8000  
Experience in years : 2  
  
7 Name : Abdallah  
Age : 29  
Salary : 7000  
Experience in years : 4  
  
8 Name : Fatma  
Age : 21  
Salary : 3000  
Experience in years : 1  
  
9 Name : Roshdy  
Age : 28  
Salary : 9000  
Experience in years : 3
```

Remove of an element of Dynamic Array hash functions

The screenshot shows a C++ IDE with a file named `HashDyArr.cpp`. The code defines a hash table using an array of pointers to a `Employee` struct. It includes functions for inserting, displaying, and removing elements. The output window shows the results of these operations, including a collision between 'Mina' and 'Mariam' at index 5. The code also demonstrates removing an element and handling a non-existent employee.

```
HashDyArr.cpp
R>>n;
e.set_Name(n);
R>>x;
e.set_Age(x);
R>>x;
e.set_Salary(x);
R>>x;
e.set_Exper(x);
DyArr.insert(e);
LinkedList.insert(e);
};
R.close();
DyArr.display();
cout << "-----";
LinkedList.display();
cout << "-----";
cout << "The collision";
cout << "The collision";
cout << "-----";

e.set_Name("Mina");
DyArr.remove(e);
DyArr.display();

e.set_Name("Kiro");
DyArr.remove(e);
```

Output:

```
1 Name : Fawzy
Age : 45
Salary : 5000
Experience in years : 8

2 Name : Yara
Age : 19
Salary : 2000
Experience in years : 0

3 Name : Ayman
Age : 33
Salary : 4000
Experience in years : 8

4 Name : Aya
Age : 26
Salary : 6000
Experience in years : 3

5 Name : Mina
Age : 30
Salary : 10000
Experience in years : 4

6 Name : Mariam
Age : 32
Salary : 8000
Experience in years : 2

7 Name : Abdallah
Age : 29
Salary : 7000
Experience in years : 4

8 Name : Fatma
Age : 21
Salary : 3000

This employee does not exist!
-----
```

Remove of an element of Linked List hash functions

```
nkedList.cpp  main.cpp
R>>n;
e.set_Name(n);
R>>x;
e.set_Age(x);
R>>x;
e.set_Salary(x);
R>>x;
e.set_Exper(x);
DyArr.insert(e);
LinkedList.insert(e);
};
R.close();
DyArr.display();
cout << "-----";
LinkedList.display();
cout << "-----";
cout << "The collision rates of line";
cout << "The collision rates of sepa";
cout << "-----";

e.set_Name("Mina");
LinkedList.remove(e);

LinkedList.display();
```

```
1 Name : Fawzy
Age : 45
Salary : 5000
Experience in years : 8

2 Name : Yara
Age : 19
Salary : 2000
Experience in years : 0

3 Name : Ayman
Age : 33
Salary : 4000
Experience in years : 8

4 Name : Aya
Age : 26
Salary : 6000
Experience in years : 3

5 Name : Mariam
Age : 32
Salary : 8000
Experience in years : 2

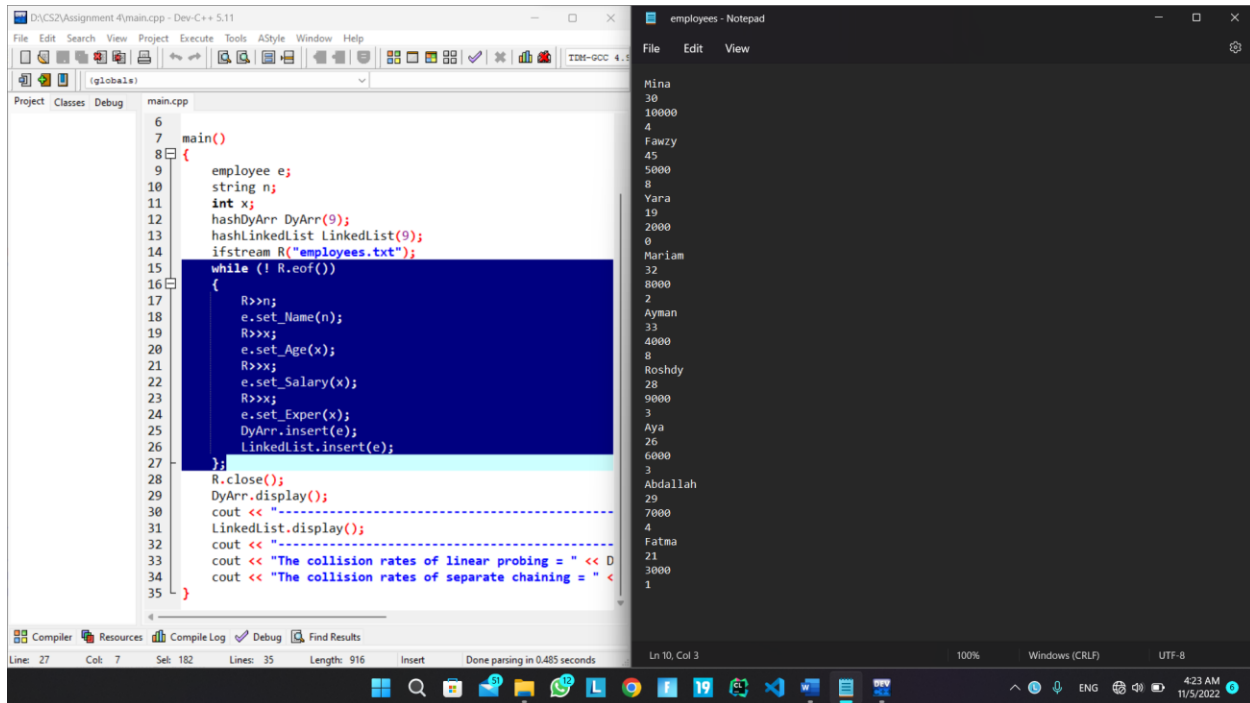
6 Name : Abdallah
Age : 29
Salary : 7000
Experience in years : 4

7 Name : Fatma
Age : 21
Salary : 3000
Experience in years : 1

8 Name : Roshdy
Age : 28
Salary : 9000
Experience in years : 3

The collision rates of separate ch
-----
This employee does not exist!
```

Bounce: implementing the data with files



## Collision rate of the two classes

```
-----
The collision rates of linear probing = 66.6667%
The collision rates of separate chaining = 55.5556%
-----
```

I think **separate chaining** is better as when a node collides, it does not take a place of other element with different hash key. As a result, that reduces the possibility of another collision.