```
/* Program for Hash Table using linear probing without replacement */ # include # include int count=0; struct hashTable { int data; int Status; /* 0 means valid data present, 1 means valid data not present in the data field */ }; struct hashTable ht[10]; void initHT() { int i; for(i=0; i<10; i++) ht[i].Status = 1; } void addData() { int data, key, i; if(count != 10) { printf("\nEnter data : "); scanf("%d", &data); key = data % 10; /*Applying hash function*/ if(ht[key].Status == 1) { ht[key].data = data; ht[key].Status = 0; printf("\nData Added to table\n"); count++; } else { for(i=0; i<10; i++) { key=(key+1)%10; if(ht[key].Status == 1) { ht[key].data = data; ht[key].Status = 0; printf("\nData Added to table\n"); count++; break; } } } } else printf("\nHash Table FULL\n"); } void delData() { int data, key, i, flag=0; if(count != 0) { printf("\nEnter data to delete : "); scanf("%d", &data); key = data % 10; /*Applying hash function*/ if(ht[key].Status == 0 && ht[key].data == data) { ht[key].Status = 1; printf("\nData deleted from table\n"); count--; } else { for(i=0; i<10; i++) { key=(key+1)%10; if(ht[key].data == data) { ht[key].Status = 1; printf("\nData deleted from table\n"); count--; flag = 1; break; } } if(flag==0) printf("\nData to be deleted not found in table\n"); } } else printf("\nHash Table EMPTY\n"); } void display() { int i; for(i=0; i <10; i++) { if(ht[i].Status == 0) { printf("| %d |", ht[i].data); } else printf("| |"); } } void main() { int ch; clrscr(); initHT(); do { printf("\nMain menu :\n"); printf("\n1) Add data \n2) Delete data \n3) Display table \n4) Exit\n\nEnter choice :"); scanf("%d", &ch); switch(ch) { case 1 : addData(); break; case 2 : delData(); break; case 3 : display(); getch(); break; case 4 : exit(0); default : printf("\nWrong choice!!"); } }while(ch != 4); }
```