Write a C program to construct hash table using hashing and collision resolution techniques.

Code:

```c
#include <stdio.h>
#define len 4

void insert(int Arr[], int data)
{
  int k = data % len;
  while (1)
  {
    if (Arr[k] == -1)
    {
      Arr[k] = data;
      return;
    }
    else
    {
      printf("A wild collision has appeared.\n");
      k++;
      if (k >= len) {
        k=0;
      }
    }
  }
}

void display(int Arr[])
{
  for (int i = 0; i < len; i++)
  {
    printf("%d | %d\n", i, Arr[i]);
  }
}

int main()
{
  int choice, value, on = 1, A[len];

  for (int i = 0; i < len; i++)
  {
    A[i] = -1;
  }

  do
  {
    printf("1. Insert\n2. Display\nChoose a operation: ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
    {
      int place = 0;
      for (int i = 0; i < len; i++)
      {
        if (A[i] == -1)
        {
          place = 1;
          break;
        }
      }

      if (place == 1)
      {
        printf("Enter data: ");
        scanf("%d", &value);
        insert(A, value);
      }
      else
      {
        printf("The Hash table is full\n");
      }
      break;
    }
    case 2:
    {
      display(A);
      break;
    }
    default:
      break;
      printf("Enter 1 to continue: ");
      scanf("%d", &on);
    }

  } while (on == 1);
}
```

Output:

```
PS C:\Users\Ajay kumar\Desktop\SEIT-B> cd "c:\Users\Ajay

1. Insert
2. Display
Choose a operation: 2

0 | -1
1 | -1
2 | -1
3 | -1

1. Insert
2. Display
Choose a operation: 1
Enter data: 3

1. Insert
2. Display
Choose a operation: 2

0 | -1
1 | -1
2 | -1
3 | 3

1. Insert
2. Display
Choose a operation: 1
Enter data: 5

1. Insert
2. Display
Choose a operation: 1
Enter data: 7

A wild collision has appeared.

1. Insert
2. Display
Choose a operation: 2

0 | 7
1 | 5
2 | -1
3 | 3
```

1. Hash the following in a table of size 12. Use any two-collision resolution technique 98, 20, 94, 27, 67, 99, 41, 0, 4, 17, 2, 15

Chaining:

0	0
1	
2	98 → 2
3	27 → 99 → 15
4	4
5	41 → 17
6	
7	67
8	20
9	
10	94
11	

0	0
1	15
2	98
3	27
4	99
5	41
6	4
7	47
8	20
9	17
10	94
11	2

Step 1: 98 % 12 = 2
Slot: 2
Step 2: 20 % 12 = 8
Slot: 8
Step 3: 94 % 12 = 10
Slot: 10
Step 4: 27 % 12 = 3
Slot: 3
Step 5: 67 % 12 = 7
Slot: 7
Step 6: 99 % 12 = 3
Collision at slot 3
Slot: 4
Step 7: 41 % 12 = 5
Slot: 5
Step 8: 0 % 12 = 0
Slot: 0
Step 9: 4 % 12 = 4
Collision at slot 4
Collision at slot 5
Slot: 6
Step 10: 17 % 12 = 5
Collision at slot 5
Collision at slot 6
Collision at slot 7
Collision at slot 8
Slot: 9
Step 11: 2 % 12 = 2
Collision at slot 2
Collision at slot 3
Collision at slot 4
Collision at slot 5
Collision at slot 6
Collision at slot 7
Collision at slot 8
Collision at slot 9
Collision at slot 10
Slot: 11
Step 12: 15 % 12 = 3
Collision at slot 3
Collision at slot 4
Collision at slot 5
Collision at slot 6
Collision at slot 7
Collision at slot 8
Collision at slot 9
Collision at slot 10
Collision at slot 11
Collision at slot 0
Slot: 1