CSE 331 – COMPUTER ORGANIZATION HOMEWORK 1 REPORT

Name – Surname : Furkan OZEV

Number : 161044036

REQUESTED:

We are asked to solve the Min-Set-Cover problem. The Min-Set-Cover problem is used to find the minimum number of sets that can cover the union of all given sets. In other words, we aim to create a union set using the minimum sets. As a result, we print the used sets.

ALGORITHM AND TO-DO:

A. The memory is allocated for storing sets.

B. The Read function is called to retrieve sets from the user.
   1. $ s0 specifies the position of sets in memory.
   2. $t2 specifies current position of sets in memory.
   3. The user is asked how many sets there will be.
   4. $t5 specifies amount of sets.
      a. It asks how many elements current set will have.
      b. $t6 specifies amount of sets elements.
         i. it asks the user for the elements of the current set
         ii. Store value in $t2 address
         iii. Replace $t2 with the next element address in memory.
      c. Replace $t2 with the next set address in memory.

C. The Readuni function is called to retrieve union set from the user.
   1. $ s0 specifies the position of sets in memory.
   2. $t2 specifies current position of union sets in memory. (add $t2, $s0, 800)
   3. It asks how many elements union set will have.
   4. $t5 specifies amount of union set.
      i. it asks the user for the elements of the current set
      ii. Store value in $t2 address
      iii. Replace $t2 with the next element address in memory.

D. The Checkfree function is called to determines union set is empty.
   1. $ s0 specifies the position of sets in memory.
   2. $ s3 specifies the position of union set in memory. (add $s3, $s0, 800)
   3. $ s1 specifies the remove element of the union set. (li $s1, '-')
   4. $ s2 specifies the end of the union set. (li $s2, '*')
   5. $a0 initilaze value is 0.
   6. $a0 specifies counter of union set.
      i. If element is equal $s2, exit function
      ii. If element is equal $s1, turn back of loop
      iii. Else increment $a0
      iv. Turn back loop

E. If $a0 is 0, it show that union set is empty. (beq $a0, 0, exitloopmain)
   1. If set is empty, exit program, else continue progress.

F. The Clear function is called to clears the memory that holds the intersection amounts for reuse.
    1. $ s1 specifies the position of sets in memory. Then, (add $s1, $s1, 1600)
    2. $s1 specifies the position of intersection amounts set in memory.
    3. $t0 and $t1 initilaze value is 0. $t1 for counter, $t0 for storing.
        i. If $t1 equal 10, exit function
        ii. Assigns a value of $t0 to each address in memory.
        iii. Replace $s1 with the next element address in memory.
        iv. Increment $t1 by 1.

G. The Intersection function is called to calculate new intersection amount.
    1. Open stack pointer. (addi $sp,$sp, -4)
    2. Add return value $ra in stack. (sw $ra, 0($sp))
    3. The lenx function is called to calculate length of union set.
        i. $ s0 specifies the position of sets in memory.
        ii. $ s3 specifies the position of union set in memory. (add $s3, $s0, 800)
        iii. $ s2 specifies the end of the union set. (li $s2, '*')
        iv. $a0 initilaze value is 0.
        v. $a0 specifies counter of union set.
            a. If element is equal $s2, exit function
            b. Else increment $a0
            c. Turn back loop
    4. $t5 specifies length of union set. (move $t5, $a0)
    5. Load return value from stack. (lw $ra,0($sp))
    6. Close stack. (addi $sp,$sp, 4)
    7. $ s0 specifies the position of sets in memory. (la $s1, Set)
    8. $ s3 specifies the position of union set in memory. (add $s3, $s1, 800)
    9. $ s2 specifies the position of intersection amount set in memory. (add $s2, $s1, 1600)
    10. For each set. (loop turns 10 times.)
        i. For each element of union set. (loop turns $a0 times)
            a. For each element of current set. (loop turns 20 times)
                1. If element of union set and element of current set are equal
                    i. İncrement amount in current $s2 value
                2. Next element address of current set. (add $s5, $s5, 4)
                3. Turn back loop.
            b. Next element address of union set. (add $s4, $s4, 4)
            c. Turn back loop.
        ii. Next set address. (add $s1, $s1, 80)
        iii. Next intersection memory address for next set. (add $s2, $s2, 4)
        iv. Turn back loop.

H. The Max function is called to determine highest intersection set.
    1. $ s1 specifies the position of sets in memory. Then, (add $s1, $s1, 1600)
    2. $s1 specifies the position of intersection amounts set in memory.
    3. $t0 specifies values in max intersection value. İnitilaze value is first value in memory.
    4. For each element. (loop turns 10 times.)
        i. $t2 specifies values in current intersection value.

ii. İf $t2 bigger than $t0
    a. Replace $t0 with the $t2.
    b. Replace $a0 current index
iii. Replace $s1 with the next element address in memory.
iv. Loop turn back.

5. $a0 will be highest intersection set number. And fuction return $a0.

İ. The Addl function is called to print found set is on the screen and store in memory.
1. Print founded set number.
2. $s1 specifies intersection set in memory. (add $s1, $s1, 2000)
3. Calculate position for this set. (sll $t2, $t1, 2)
4. And store it in this address position.

J. The Removelem function is called deletes element of founded set from union set.
1. $t1 specifies number of set to delete from union set. (move $t1, $s7)
2. $s1 specifies address position of sets in memory. (la $s1, Set)
3. $s2 specifies address position of union set. (add $s2, $s1, 800)
4. Find the location of the set in memory to be deleted from the union set.
5. The function checks whether each element of the set to be deleted is in the union set.
    i. If the element exists in the union set, that element is deleted from the union set.

K. Turn mainloop back.

EXTRAS:

1. The sets were taken sequentially from the console.
2. Elements in sets have no limits.
3. By changing some fixed numbers over the code, we can increase the maximum number of sets and the number of elements in a set.

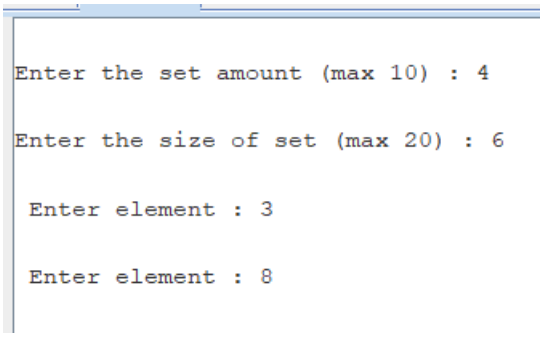
TEST CASES:

CASE 1.

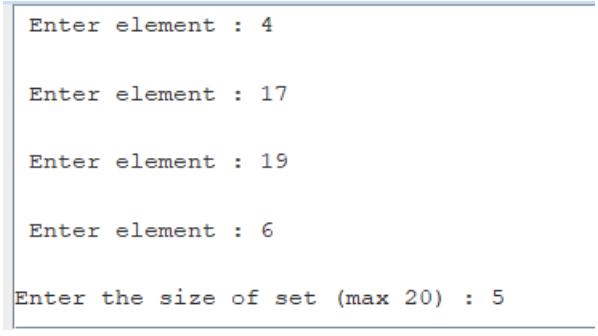
```
SET1: 3 8 4 17 19 6
SET2: 5 9 12 6 4
SET3: 2 7 13 6 5
SET4: 17 3 13 2

UNION: 3 13 5 12
```

1. SS



2. SS



3. SS

```
Enter element : 5

Enter element : 9

Enter element : 12

Enter element : 6
```

4. SS

```
Enter element : 4

Enter the size of set (max 20) : 5

Enter element : 2

Enter element : 7

Enter element : 13
```

5. SS

```
Enter element : 6

Enter element : 5

Enter the size of set (max 20) : 4

Enter element : 17
```

6. SS

```
Enter element : 3

Enter element : 13

Enter element : 2

Enter size of union set : 4

Enter element : 3
```

7. SS

```
Enter element : 13

Enter element : 5

Enter element : 12

Result Sets : 2,4,
-- program is finished running --
```

RESULT SETS : 2 , 4

CASE 2:

```
SET1: 3 4
SET2: 2 4 6
SET3: 2 3 6
SET4: 6 8
SET5: 3 5 6

UNION: 2 3 5 6 8
```

1.SS

```
Enter the set amount (max 10) : 5

Enter the size of set (max 20) : 2

 Enter element : 3

 Enter element : 4
```

2.SS

```
Enter the size of set (max 20) : 3

 Enter element : 2

 Enter element : 4

 Enter element : 6
```

3.SS

```
Enter the size of set (max 20) : 3

 Enter element : 2

 Enter element : 3

 Enter element : 6
```

5.SS

```
Enter the size of set (max 20) : 2

 Enter element : 6

 Enter element : 8
```

6.SS

```
Enter the size of set (max 20) : 3

 Enter element : 3

 Enter element : 5

 Enter element : 6
```

7.SS

```
Enter size of union set : 5

Enter element : 2

Enter element : 3

Enter element : 5
```

8.SS

```
 Enter element : 6

 Enter element : 8

 Result Sets : 3,4,5,
-- program is finished running --
```

RESULT SETS : 3 , 4 , 5