**PROJECT**

**Name:** Akshansh Manchanda, Shivanshu Dhawan Anil

**Registration Number:** 14BCE0579, 14BCE0580

**Title:** Fixtures of Single Elimination Tournament

**Course Name:** Data Structures & Algorithms

**Course Code:** CSE220

**Semester:** Fall 2015

**Problem Statement**: Write a program to fix matches (single elimination tournament) for a list of participants entered by user using a circular queue and also delete any duplicates by name.

**Software Used:**  Dev C++ Version 5.11 <http://orwelldevcpp.blogspot.in/>

**Functions / Modules:**

Note: Front Index = f % size, Rear Index = r % size and we consider indexes 0 to size-1 arranged in a circle when we refer to next index(index next to size-1 is 0). CQ – Circular Queue, size – defined as 100 (max size of CQ)

1. void ini( ): Called to initialize CQ as empty.

LOGIC:

Front Index = -1

Rear Index = -1

1. int isfull( ): Called to check if circular queue is full.

LOGIC:

If ( Index next to Rear Index ) equals to Front Index

Return the Value 1 (int)

Else

Return the Value 0 (int)

1. int isempty( ): Called to check if circular queue is empty

LOGIC:

If Rear Index equals to -1

Return the Value 1 (int)

Else

Return the Value 0 (int)

1. void enqueue(char\*) : Called to enqueue a string to CQ

LOGIC:

Character Pointer pointing to a String is taken as a Parameter

If CQ is full

Display “Full”

Else

If CQ is empty

Front Index = 0

Rear Index = 0

If CQ is partially filled

Rear Index = Index Next To It

For i = 0, Till \*(str + i) does not equal to ‘\0’, i = i +1

name[Rear Index][i] = \*(str + i)

name[Rear Index][i] = ‘\0’

1. char\* remove( ) : Called to remove a string

LOGIC:

If CQ is empty

Display “Empty”

p(string)=”\0”(Return String is Empty)

If CQ has only 1 string

Copy into p, name[Front Index]

CQ should be empty after deletion so

Front Index = -1

Rear Index = -1

If CQ has more than 1 string

Copy into p, name[Front Index]

Front Index = Index Next To It

Return the string pointer p

1. void display( ) : Called to display tournament status

LOGIC:

Display “Tournament Status”

If Rear Index is greater than or equal to Front Index

For i = Front Index, Till i <= Rear Index, i = i + 1

Display name[i]

Else

For i = Front Index, Till i < size, i = i + 1

Display name[i]

For i = 0, i <= Rear Index, i = i + 1

Display name[i]

1. void tournament( ) : Called to give byes, generate tournament status after each round and decide winner

LOGIC:

Display “Tournament Initiated”

Display Tournament Status

Store Rear Index in rprv

If Front Index is less than or equal to Rear Index

Size of CQ = Rear Index – Front Index + 1

Else

Size of CQ = size – Front Index + Rear Index + 1

pwt = 2^0

While pwt is less than Size of CQ

pwt = 2 x pwt

Now pwt is the least value of 2^n greater than or equal to Size of CQ

No of Byes = 2^n – Size of CQ

Counter=0

If there are byes to give

Display “Bye given to:”

While Counter is not equal to No of Byes

Display name[Front Index]

Remove this string and enqueue it

Counter = Counter + 1

While Front Index does not equal to Index Next To rprv

Remove a string and store in cmp1

Remove a string and store in cmp2

Do

Take in ch

If ch equals to 1

Enqueue cmp1

Else if ch equals to 2

Enqueue cmp2

Else

Display “Wrong Input”

While ch is not equal to 1 and 2

If CQ has only one string

Size of CQ was 2

Break while loop

While Size of CQ was not 2

Display Tournament Status

Store Rear Index in rprv

While Front Index does not equal to Index Next To rprv

Remove a string and store in cmp1

Remove a string and store in cmp2

Do

Take in ch

If ch equals to 1

Enqueue cmp1

Else if ch equals to 2

Enqueue cmp2

Else

Display “Wrong Input”

While ch is not equal to 1 and 2

If CQ has only one string

Size of CQ was 2

Break while loop

Display The Winner as name[Front Index]

1. int\* chkdup( ) : Called to return information on duplicates in CQ

LOGIC:

i, j indices of two strings for comparison

Assumed Duplicate Found = False (dupfnd[0]=0)

Assumed Position of Duplicate = 0 (dupfnd[1]=0)

If Rear Index is greater than or equal to Front Index

For i = Front Index, Till i <= Rear Index, i = i + 1

For j = i + 1, Till j <= Rear Index, j = j + 1

If name[i] and name[j] are equivalent

Duplicate Found = True

Position of Duplicate = j

Break for loop

If Duplicate was found

Break for loop

Else

For i = Front Index, Till i < size, i = i + 1

For j = i + 1, Till j < size; j = j + 1

If name[i] and name[j] are equivalent

Duplicate Found = True

Position of Duplicate = j

Break for loop

If Duplicate was found

Break for loop

For j = 0, Till j <= Rear Index; j = j + 1

If name[i] and name[j] are equivalent

Duplicate Found = True

Position of Duplicate = j

Break for loop

If Duplicate was found

Break for loop

For i = 0, Till i <= Rear Index, i = i + 1

For j = 0, Till j <= Rear Index; j = j + 1

If name[i] and name[j] are equivalent

Duplicate Found = True

Position of Duplicate = j

Break for loop

If Duplicate was found

Break for loop

Return Info on Duplicate Found as integer array dupfnd

1. void deldup( ) : Called to delete all duplicates in CQ

LOGIC:

Check for duplicates

If any duplicate was found

Display “Duplicates found”

Do

While Front Index does not equal to \*(dpf+1)

Or Index of Duplicate Found

Remove a string and enqueue it

Remove a string (Duplicate)

Check for duplicates

While duplicates were found

Else

Display “No duplicates found”

All functions above are in scope of and are members of class cq (CIRCULAR QUEUE)

**List of Errors Encountered while coding the project:**

* Case of Two Participants/Last Case:

“while((f%size)!=((rprv+1)%size))”

This statement breaks for all rounds except the final round where there are only two participants. Earlier in case of two participants, after removal of both participants the CQ became empty and so the Front and Rear Index were given the value -1.

After this one string is enqueued which is the winner, but the condition of while loop above is still satisfied and the tournament is continued even though it is over.

To rectify this, we added new conditions for the last case.

**Key or challenging logic in the project:**

In the function tournament( ) and deldup( ) we required access to strings that were neither at the Front Index nor at the Rear Index. To solve this problem we made changes to the functions enqueue( ) and remove( ).

Enqueue takes in a character pointer that points to the string to be enqueued and Remove returns the string pointer of a string that keeps a copy of the string that is removed.

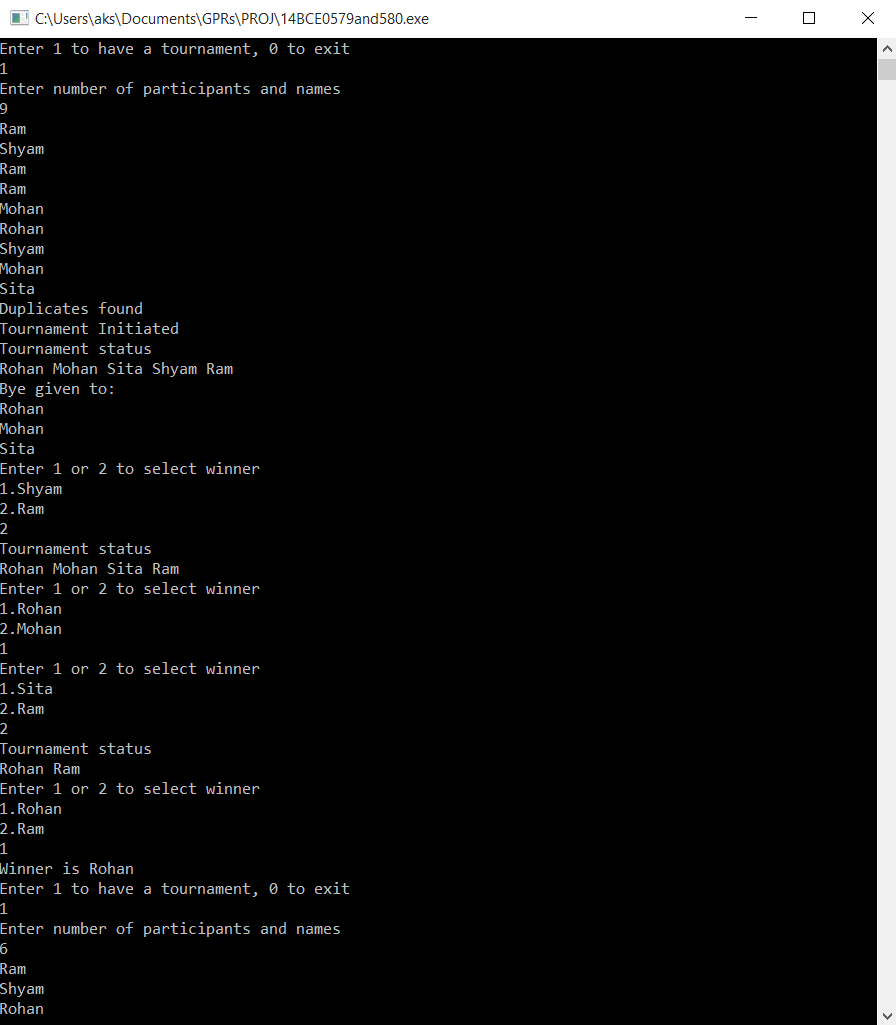
“enqueue(remove())”

To access any string neither at Front Index nor at Rear index we keep on performing above operation till we reach the desired index.

**Complete Code**

Code attached “14BCE0579and580.cpp”

**Elaborate Sample Input and Output**

****

