Name:Jacob V Sanoj	SRN:PES1UG20EC083	Section:F1
	Date:28-06-2021	Week Number:7

1 Define a structure called cricket that will describe the following information:

player name

team name

batting average

Using cricket, declare an array player with 5 elements and write a program to read the information about all the 5 players and print a team-wise list containing names of player with their batting average.Write functions for the following:

i) Read the information of all the 5 players

ii)Sorting the players

iii)Displaying team-wise list containing names of player with their batting average

Input:

Enter data of 5 players

Enter PName TName BAvg for player-1 = sachin

India

98

Enter PName TName BAvg for player-2 = Rahul

India

45

Enter PName TName BAvg for player-3 = Jonty

Australia

89

Enter PName TName BAvg for player-4 = Imran

pakistan

75

Enter PName TName BAvg for player-5 = Shen

Australia

29

Output:

After teamwise sorting... Player list is

Jonty	Australia	89.00
Shen	Australia	29.00
sachin	India	98.00
Rahul	India	45.00
Imran	pakistan	75.00

Program:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

```
struct cricket
{
    char playername[20];
    char teamname[20];
    int average;
};

int main()
{
    struct cricket player[5];
    for (int i = 0; i < 5; i++)
    {
        printf("Enter the name of the player\n");
        scanf("%s", player[i].playername);
        printf("Enter the name of the team\n");
        scanf("%s", player[i].teamname);
        printf("Enter the average\n");
        scanf("%d", &player[i].average);
    }

    int pos;

    for (int i = 0; i < 4; i++)
    {
        pos = i;
        for (int j = i; j < 4; j++)
        {
            if (strncmp(player[i].teamname, player[j].teamname,
strlen(player[i].teamname)) == 0)
            {
                struct cricket temp;
                temp = player[pos + 1];
```

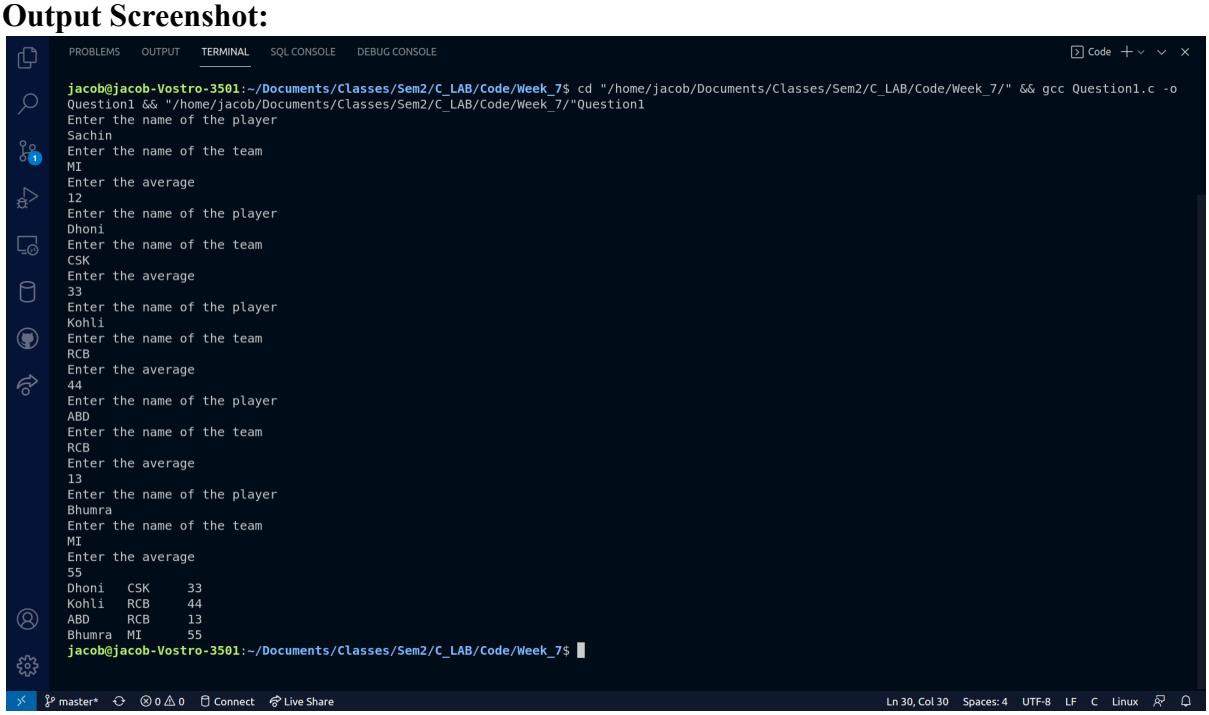
```
                player[pos + 1] = player[j];
                player[j] = temp;
                pos = pos + 1;
            }
        }
    }

    for (int i = 0; i < 4; i++)
    {

        printf("%s\t%s\t%d\n", player[i].playername,
player[i].teamname, player[i].average);
    }

}
```

Output Screenshot:



2	**Implement Priority Queue using an Unordered Linked list.** Write functions for the following 1)Initialization 2)Enqueue 3)Dequeue 4)Display **Output:** enter ua choice 1.insert 2.delete 3.display 4 exit 1 enter the detail and priority 10 1 enter ua choice 1.insert 2.delete 3.display 4 exit 1 enter the detail and priority 20 2 enter ua choice 1.insert 2.delete 3.display 4 exit

```
1
enter the detail and priority
30
3
enter ua choice
1.insert 2.delete 3.display 4 exit
3
30 3
20 2
10 1
enter ua choice
1.insert 2.delete 3.display 4 exit
1
enter the detail and priority
40
0
enter ua choice
1.insert 2.delete 3.display 4 exit
3
40 0
30 3
```

```
20 2
10 1
enter ua choice
1.insert 2.delete 3.display 4 exit
2
deleted node detail is 30 with priority 3
enter ua choice
1.insert 2.delete 3.display 4 exit
2
deleted node detail is 20 with priority 2
enter ua choice
1.insert 2.delete 3.display 4 exit
2
deleted node detail is 10 with priority 1
enter ua choice
1.insert 2.delete 3.display 4 exit
2
deleted node detail is 40 with priority 0
enter ua choice
1.insert 2.delete 3.display 4 exit
2
```

no elements to delete

enter ua choice

1.insert 2.delete 3.display 4 exit

4

Program:

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
struct component
{
    char details[100];
    int priority;
};
struct node
{
    struct component compo;
    struct node *link;
};
struct queue
{
    struct node *head;
};
struct node *high(struct queue *p)
{
    int priority_value = ((p->head)->compo).priority;
    struct node *temp;
    temp = p->head;
    while (temp != NULL)
    {
        if ((temp->compo).priority > priority_value)
```

```
        {
            priority_value = (temp->compo).priority;
        }
        temp = temp->link;
    }
    temp = p->head;
    while (1)
    {
        if ((temp->compo).priority == priority_value)
        {
            break;
        }
        temp = temp->link;
    }
    return temp;
}
struct node *low(struct queue *p)
{
    struct node *temp;
    int priority_value = ((p->head)->compo).priority;
    temp = p->head;
    while (temp != NULL)
    {
        if ((temp->compo).priority < priority_value)
        {
            priority_value = (temp->compo).priority;
        }
        temp = temp->link;
    }
    temp = p->head;
    while (1)
    {
```

```
        if ((temp->compo).priority == priority_value)
        {
            break;
        }
        temp = temp->link;
    }
    return temp;
}
void enqueue(struct queue *p)
{
    char det[20];
    int priority;
    printf("Enter the details of the newnode\n");
    scanf("%s", det);
    printf("Enter the priority of the newnode\n");
    scanf("%d", &priority);
    struct node *temp;
    struct node *newnode;
    struct node *new;
    new = low(p);
    newnode = (struct node *)malloc(sizeof(struct node));
    temp = (struct node *)malloc(sizeof(struct node));
    temp->link = p->head;
    while (temp->link != low(p))
    {
        temp = temp->link;
    }
    int k = 0;
    if (temp->link == p->head)
    {
        k = 1;
    }
```

```
    temp->link = newnode;
    newnode->link = new;
    strcpy((newnode->compo).details, det);
    (newnode->compo).priority = priority;
    if (k == 1)
    {
        p->head = newnode;
    }
}
void dequeue(struct queue *p)
{
    struct node *temp;
    temp = (struct node *)malloc(sizeof(struct node));
    temp->link = p->head;
    while (temp->link != high(p))
    {
        temp = temp->link;
    }
    struct node *new;
    new = high(p);
    int k = 0;
    if (temp->link == p->head)
    {
        k = 1;
    }
    if (k == 1)
    {
        p->head = new->link;
    }
    else
    {
        temp->link = new->link;
```

```
    }
}
void display(struct queue *p)
{
    struct node *temp;
    temp = p->head;
    while (temp != NULL)
    {
        printf("%s %d \n", (temp->compo).details,
(temp->compo).priority);
        temp = temp->link;
    }
}
void init(struct queue *p, char head_set[100], int
head_priority)
{
    struct node *temp;
    temp = (struct node *)malloc(sizeof(struct node));
    p->head = temp;
    strcpy((temp->compo).details, head_set);
    (temp->compo).priority = head_priority;
}
int main()
{
    struct queue *p, queue1;
    p = &queue1;
    char s[100];
    int n;
    printf("Enter the details of the head of the queue\n");
    scanf("%s", s);
    printf("Enter the priority of the head\n");
    scanf("%d", &n);
```

```
    init(p, s, n);
    char choice[20];
    printf("Enter your choice.\n");
    printf("1.enqueue\n");
    printf("2.dequeue\n");
    printf("3.display\n");
    printf("4.exit\n");
    scanf("%s", choice);
    while (1)
    {
        if (strncmp(choice, "enqueue", 7) == 0)
        {
            enqueue(p);
        }
        else if (strncmp(choice, "dequeue", 7) == 0)
        {
            dequeue(p);
        }
        else if (strncmp(choice, "display", 7) == 0)
        {
            display(p);
        }
        else if (strncmp(choice, "exit", 4) == 0)
        {
            break;
        }
        else
        {
            printf("invalid choice");
            printf("Enter your choice.\n");
            printf("1.enqueue\n");
            printf("2.dequeue\n");
```

```
            printf("3.display\n");
            printf("4.exit\n");
            scanf("%s", choice);
        }
        printf("Enter your choice.\n");
        printf("1.enqueue\n");
        printf("2.dequeue\n");
        printf("3.display\n");
        printf("4.exit\n");
        scanf("%s", choice);
    }
}
```

Output Screenshot:

```
PROBLEMS   OUTPUT   TERMINAL   SQL CONSOLE   DEBUG CONSOLE                    Code

Enter the details of the head of the queue
darkness
Enter the priority of the head
2
Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
enqueue
Enter the details of the newnode
hello
Enter the priority of the newnode
1
Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
enqueue
Enter the details of the newnode
smile
Enter the priority of the newnode
3
Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
display
smile 3
hello 1
darkness 2
Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
dequeue
Enter your choice.
```

```
PROBLEMS    OUTPUT    TERMINAL    SQL CONSOLE    DEBUG CONSOLE                    Code

Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
enqueue
Enter the details of the newnode
smile
Enter the priority of the newnode
3
Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
display
smile 3
hello 1
darkness 2
Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
dequeue
Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
display
hello 1
darkness 2
Enter your choice.
1.enqueue
2.dequeue
3.display
4.exit
exit
jacob@jacob-Vostro-3501:~/Documents/Classes/Sem2/C_LAB/Code/Week_7$
```