Week 7

ANIRUDH ROWJEE `PES2UG20CS050`

Task 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:

1. Read the information of all the 5 players
2. Sorting the players by their team name
3. Displaying team-wise list containing names of player with their batting average

`main.c`

```c
#include <stdio.h>
#include "utils.h"

int main()
{

    int count;

    printf("Enter the number of players >> ");
    scanf("%d", &count);

    // allocate space for all the players
    // read the user data for the players
    cricket_t* player_array = generate_player_array(count);

    // sort the player array on the basis of
    sort_player_array(count, player_array);

    // display the sorted player array
    display_player_array(count, player_array);

    // free the array of players
    free_player_array(player_array);
    return 0;
}
```

`utils.h`

```c

// header file for Player Operations
// define the shared structures
typedef struct cricket {
    char name[30];
    char team_name[30];
    float batting_average;
} cricket_t;


// load the players into an array and return the pointer to the array
cricket_t* generate_player_array(int count);

// free the array generate
int free_player_array(cricket_t* player_array);

// sort the array
void sort_player_array(int count, cricket_t* player_array);

// display the array
void display_player_array(int count, cricket_t* player_array);
```

`utils.c`

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "utils.h"



// load the players into an array and return the pointer to the array
cricket_t* generate_player_array(int count)
{
    cricket_t* cricketer_array = (cricket_t*)malloc(sizeof(cricket_t)*count);

    for (int i = 0; i < count; i++)
    {
        printf("Player %d >> \n", i+1);
        printf("Player Name >> ");
        scanf("%s", cricketer_array[i].name);
        printf("Player Team Name >> ");
        scanf("%s", cricketer_array[i].team_name);
        printf("Player Batting Average >> ");
        scanf("%f", &cricketer_array[i].batting_average);
    }

    return cricketer_array;
}

// free the array generate
int free_player_array(cricket_t* player_array)
{
    free(player_array);
    player_array = NULL;
    return 0;
}

// sort the array
void sort_player_array(int count, cricket_t* player_array)
{
    // implement bubble sort based on strcmp;
    for (int i = 0; i < count; i++)
    {
        for (int j = 0; j < count - i - 1; j++)
        {
            if (
                strcmp(
                    player_array[j].team_name,
                    player_array[j+1].team_name
                ) >= 1
            )
            {
                cricket_t temp;
                temp = player_array[j];
                player_array[j] = player_array[j+1];
                player_array[j+1] = temp;
            }
        }
    }
}

// display the array
void display_player_array(int count, cricket_t* player_array)
{
    printf("\n----- PLAYERS -----\n");
    for (int i = 0; i < count; i++)
    {
        printf(
                "%s | %s | %f\n",
                player_array[i].name,
                player_array[i].team_name,
                player_array[i].batting_average
                );
    }
    printf("-------------------\n");
}
```

Console

```
nidavellir :: UE20CS152/week6/t1 » ./main
Enter the number of players >> 5
Player 1 >>
Player Name >> Sachin
Player Team Name >> India
Player Batting Average >> 98
Player 2 >>
Player Name >> Rahul
Player Team Name >> India
Player Batting Average >> 45
Player 3 >>
Player Name >> Jonty
Player Team Name >> Australia
Player Batting Average >> 89
Player 4 >>
Player Name >> Imran
Player Team Name >> Pakistan
Player Batting Average >> 75
Player 5 >>
Player Name >> Shen
Player Team Name >> Australia
Player Batting Average >> 29

----- PLAYERS -----
Jonty | Australia | 89.000000
Shen | Australia | 29.000000
Sachin | India | 98.000000
```

```
Rahul | India | 45.000000
Imran | Pakistan | 75.000000
-------------------
nidavellir :: UE20CS152/week6/t1 »
```

Task 2

Implement a Priority Queue using Unoredered Linked Lists.

Write Functions To

1. Initialize
2. Enqueue
3. Dequeue
4. Display

`main.c`

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

typedef struct node {
    int priority;
    int data;
    struct node* next;
} node_t;

node_t* spawn_element()
{
    /*
     * function to create a new node and accept the priority value
     * as well as the Data value from the user
     */
    node_t* new_node = (node_t*)malloc(sizeof(node_t));

    printf("Enter the Data Value >> ");
    scanf("%d", &new_node->data);
    printf("Enter the Data Priority >> ");
    scanf("%d", &new_node->priority);
    new_node->next = NULL;
    return new_node;
}

// make the queue
node_t* initialize_pqueue()
{
    /*
     * function to initalize the priority queue with the first
     * element.
     */
    return spawn_element();
}


int main()
{
    int choice = 0;
    printf("Welcome To The Priority Queue Manager!\n");

    node_t* HEAD = NULL;
    node_t* temp = HEAD;

    while (choice != 4)
    {
        temp = HEAD;
        printf(
            "\nPick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.\n"
        );
        scanf("%d", &choice);

        switch (choice)
        {

            case 1:
            {
                // function to print the priority queue

                if (temp == NULL)
                {
                    printf("The List is Empty!\n");
                    break;
                }

                printf("\nPriority Queue >> \n");
                while (temp != NULL)
                {
                    printf("D: %d | P: %d\n", temp->data, temp->priority);
                    temp = temp->next;
                }
                break;
            }

            case 2:
            {
                // insert based on highest priority first
                node_t* new_node;
                new_node = spawn_element();
                // basic null check - if head is null, just add the new element
                if (temp == NULL)
                {
                    HEAD = new_node;
                    break;
                }
                if (temp == HEAD)
                {
                    if (new_node->priority > temp->priority)
                    {
                        new_node->next = temp;
                        HEAD = new_node;
                        temp = HEAD;
                        break;
                    }
                    else {
                        // this loop will terminate on the last element
                        // if it hasn't been exited by then, we can assume
                        // the new node has the smallest value
                        while (temp->next != NULL)
                        {
                            // if this is true, we insert new_node before temp->next;
                            if (new_node->priority > temp->next->priority)
                            {
                                new_node->next = temp->next;
                                temp->next = new_node;
                                break;
                            }
                            else {
                                // check to see if we're at the last element - if so,
                                // we allocate the current element either befo
                                temp = temp->next;
                            }
                        }
                        // upon the exit without return, we will assign
                        // new_node to temp->next
                        temp->next = new_node;
                    }
                }
                break;
            }

            case 3:
            {
                // function to remove the element at HEAD since we
                // inserted in sorted order based on priority
                if (temp == NULL)
                {
                    printf("Cannot Dequeue An Empty Priority Queue!\n");
                    break;
                }
                else {
                    node_t* runner = temp->next;
                    printf("Dequeueing (%d, %d)\n", temp->priority, temp->data);
                    free(temp);
                    // temp = NULL;
                    HEAD = runner;
                }
                break;
            }

            case 4:
            {
                // Function to cleanup the allocated memory
                if (HEAD == NULL)
                {
                    printf("The PQueue is Empty!\n");
                }
                node_t* runner = temp;
                while (temp != NULL)
                {
                    runner = temp->next;
                    free(temp);
                    temp = NULL;
                    temp = runner;
                }
                printf("Thank you for using the Linked List Manager!\n");
                break;
            }

            default:
                printf("Invalid Option!\n");
                break;
        }

    }

    return 0;
}
```

Console

```
nidavellir :: UE20CS152/week6/t2 » ./main
Welcome To The Priority Queue Manager!

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
1
The List is Empty!

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
2
Enter the Data Value >> 10
Enter the Data Priority >> 15

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
2
Enter the Data Value >> 12
Enter the Data Priority >> 17

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
2
Enter the Data Value >> 11
Enter the Data Priority >> 16

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
1

Priority Queue >>
D: 12 | P: 17
D: 11 | P: 16
D: 10 | P: 15

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
2
Enter the Data Value >> 19
Enter the Data Priority >> 14

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
2
Enter the Data Value >> 77
Enter the Data Priority >> 10

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
2
Enter the Data Value >> 56
Enter the Data Priority >> 13

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
1

Priority Queue >>
D: 12 | P: 17
D: 11 | P: 16
D: 10 | P: 15
D: 19 | P: 14
D: 56 | P: 13
D: 77 | P: 10

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
3
Dequeueing (17, 12)

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
3
Dequeueing (16, 11)
```

```
Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
3
Dequeueing (15, 10)

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
3
Dequeueing (14, 19)

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
3
Dequeueing (13, 56)

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
3
Dequeueing (10, 77)

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
3
Cannot Dequeue An Empty Priority Queue!

Pick 1 to display, 2 to enqueue, 3 to dequeue, and 4 to exit.
4
The PQueue is Empty!
Thank you for using the Linked List Manager!
nidavellir :: UE20CS152/week6/t2 »
```