ASSIGNMENT 2

Name: Vanaj Kamboj
Roll No: 21355100

Task 1
My Code:

```
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
#include "t1.h"

void printArray(int* arr, int size){
  int i;
  for(i=0; i<size;i++){
    printf("%d ", arr[i]);
  }
  printf("\n");
}
//Fills the array with ascending, consecutive numbers, starting from 0.
void fill_ascending(int *array, int size)
{
    //TODO:
    for(int i = 0; i < size; i++)
    {
        array[i] = i+1;
    }
}
//Fills the array with descending numbers, starting from size-1
void fill_descending(int *array, int size)
{
    //TODO:
    for(int i = 0; i < size; i++)
    {
       array[i] = size-i+1;
    }
}

//Fills the array with uniform numbers.
void fill_uniform(int *array, int size)
{
    //TODO:
    int random = rand()%size;
    for(int i = 0; i < size; i++)
    {
        array[i] = random;
    }
}
```

```
//Fills the array with random numbers within 0 and size-1. Duplicates are allowed.
void fill_with_duplicates(int *array, int size)
{
    //TODO:

    for(int i = 0; i < size; i++)
    {
        array[i] = rand()%size;
    }
    // printArray(array,7);

}
//Fills the array with unique numbers between 0 and size-1 in a shuffled order.
Duplicates are not allowed.
void fill_without_duplicates(int *array, int size)
{
    //TODO
    int i = 0;
    while (i < size)
    {
        array[i] = i;
        i++;
    }

    if (size > 1)
    {
        i = 0;
        while (i < size - 1)
        {
            int j = i + rand() / (RAND_MAX / (size - i) + 1);
            int t = array[j];
            array[j] = array[i];
            array[i] = t;
            i++;
        }
    }
}
```

Task 2&3

```
#include <stdio.h>
#include "t2.h"

int number_comparisons = 0;
int number_swaps = 0;

void selectionSort(int arr[], int size)
{
    int i,j;
    for(i=0;i<size-1;i++){
        int min=i;
        // number_comparisons++;
        for(j=i+1;j<size;j++){
            if(arr[j]<arr[min]){
            number_comparisons++;
            min=j;
            }
        }
        if(min!=i){
            // swap(&arr[i],&arr[min]);
            int temp=arr[i];
            arr[i]=arr[min];
            arr[min]=temp;

            number_swaps++;
        }
    }
}

void insertionSort(int arr[], int size)
{
    int i,j;
  for(i=1;i<size;i++){
      int temp=arr[i];
      number_comparisons++;
      for(j=i-1;j>=0&&arr[j]>temp;j--){
          arr[j+1]=arr[j];
          number_swaps++;
      }
      arr[j+1]=temp;
  }

}

// The implementation idea for quickSort function was taken from
// https://stackoverflow.com/questions/50858823/c-quicksort-in-one-function-with-2-
parameters-int-length

void quickSort(int* array, int size)
{
        int Pivot = array[size-1];
        int idxLargestElement = 0;
        for (int i = 0; i < size-1; i++)
        {number_comparisons++;
                if (array[i] < Pivot)
                {
                        // Swap largest element with this
                        number_swaps++;
                        int temp = array[i];
                        array[i] = array[idxLargestElement];
                        array[idxLargestElement]  = temp;
                        idxLargestElement++;
                }
        }
        // swap pivot with array[idxLargestElement]
        number_swaps++;
        int temp = array[idxLargestElement];
        array[idxLargestElement] = array[size-1];
        array[size-1] = temp;
        if (idxLargestElement > 1)
                quickSort(array, idxLargestElement);
        if (size-idxLargestElement-1 > 1)
                quickSort(array+idxLargestElement+1, size-idxLargestElement-1);
}
```

Task 4
Code:

```
#include <stdio.h>
#include <string.h>
#include <curses.h> //conio doesn't work for macOS
#include <stdlib.h>

#define MAX_BUFFER 256
#define MAX_ROWS 18626

const int elementSize = 256;

// title,platform,Score,release_year

// Struct for storing a Pokemon's details
typedef struct Game
{
    char *title;
    char *platform;
    int score;
    int year;
} Game;

Game *getGame(char buffer[])
{
    // printf("\n%s\n", buffer);
    int column = 0;
    int foundQuote = 0;
    char element[elementSize];
    memset(element, 0, elementSize); // Clear variable to build next field

    // Result Game struct
```

```c
    Game *game = malloc(sizeof(Game));

    // Iterate over the buffer and print column elements
    for (int i = 0; i < strlen(buffer); i++)
    {
        switch (buffer[i])
        {
        case ',':
            if (!foundQuote)
            {

                // Column data present in element
                // printf("Column data -> %s, Found Quote %d\n", element,
foundQuote);

                // Store this element in struct!
                // TODO: This part can be abstracted into a neat function...
                switch (column)
                {
                case 0:
                    game->title = strdup(element);
                    break;
                case 1:
                    game->platform = strdup(element);
                    break;
                case 2:
                    game->score = atoi(element);
                    break;
                case 3:
                    game->year = atoi(element);
                    break;
                default:
                    break;
                }
                column++;
                // Store this in our struct.
                memset(element, 0, elementSize); // Clear variable to build next
field
            }
            break;
        case '\"':
            foundQuote = !foundQuote;
            continue;
        default:
            strcat(element, (char[2]){buffer[i], '\0'});
            break;
        }
    }

    game->year = atoi(element);
    return game;
```

```c
}

void insertionSort(Game *arr[], int size)
{
    int i, j;
    for (i = 1; i < size; i++)
    {
        Game *temp = arr[i];

        for (j = i - 1; j >= 0 && arr[j]->score < temp->score; j--)
        {
            arr[j + 1] = arr[j];
        }
        arr[j + 1] = temp;
    }
}

void print_game(Game *p)
{
    printf("%-60s %-25s %-4d %-4d\n", p->title, p->platform, p->score, p->year);
}

int main(int argc, char **argv)
{
    if (argc < 2)
    {
        printf("Filename not provided. Exiting...\n");
        return 1;
    }
    char *fileName = argv[1];
    FILE *file = fopen(fileName, "r");
    Game *gameArray[MAX_ROWS - 1];
    int numGames = 0;

    if (!file)
        printf("Can not open the File\n");
    else
    {
        printf("File found! Parsing file...\n");

        char buffer[1024];
        int isFirst = 1;

        while (fgets(buffer, 1024, file))
        {
            if (isFirst)
            {
                isFirst = 0;
                continue;
            }
            gameArray[numGames] = getGame(buffer);
            numGames++;
            insertionSort(gameArray, numGames); // Sort the list
        }
    }

    for (int i = 0; i < 10; i++)
    {
        print_game(gameArray[i]);
    }

    fclose(file);

    return 0;
}
```

Discuss how you would get the top 5 games for each of the last 20 years?

Answer:
My idea is create a hash table that would use year as its key and each bucket of hash table will hold a linked list of that year of records of games using the chaining method.
After assigning values into the hash table we can sort the linked list using a sorting algorithm and then print the top 5 games of each year.