COS10007 Developing Technical Software TP 2 2022

Assignment 2 report

Name: *Hai Nam Ngo*

Student ID: *103488515*

Lab class:

1. *Tuesday / 8.30am-10.30am / TA110*
2. *Friday / 1.30pm-3.30pm / TC223*

Teacher: *Prince Kurumthodathu Surendran*

Due Date: Monday 15th May 2023 at 11:59 pm

Date Submitted: Monday 15th May 2023 at 10:00 pm

Question 1

1. **Program description**

Creating a C Program to read data from a text file which has 10 rows and 4 columns with the data type of each column is: char, int, int, double respectively.

The system allows user to choose 5 options from the menu: display the list, sort the list, search for a specific person in the list, insert array into linked list and quit the program.

For this task, bubble sort and selection sort will be used for "sort the list" task, and both linear and binary search will be used for "searching" task.

The program will contain: 2 header files and 4 C files for the main and other functions, which will make it easier to understand.

2. **Inputs and Outputs**

The inputs and outputs for this program are described in Table 1.

Table 1. Data dictionary:

Data to be stored	Sample data	Type of data	C type	Input method	In / Out	Variable name
Player name	"Kent"	text: maximum 10 characters	string	scanf fscanf	printf	Player
Age	23	real number	Int	scanf	printf	Age

				fscanf		
Rank	25	real number	Int	scanf fscanf	printf	Rank
Score	23.540000	real number	double	scanf fscanf	Printf	score
Menu option	1	number	int	scanf	functio n	option
Sort option	1	number	int	scanf	functio n	Sort_option
Sort order	asc	text	string	scanf	functio n	Sort_order
Player to search	“Kent”	text: maximum 10 characters	string	scanf	functio n	Search_key
Search option	1	number	int	scanf	functio n	Search_option

3. Algorithm

Program steps:

3. Declare the necessary variables and structures, including the option variable, the info array structure, and the file pointer.

4. Open the file in reading mode and check if the file exists.

5. Use a loop to read the data from the file and store it in the info array.

6. Display the menu and prompt the user to select an option.

7. Use a switch statement to execute the corresponding function based on the user's option:

- If the user selects option 1, call the displayFile function to display all the players' details.
- If the user selects option 2, call the sort function to sort the players' details and then call the displayFile function to display the sorted details.

- If the user selects option 3, call the searching function to search for a specific player's details.

- If the user selects option 4, call the linked_list function to insert the array into a linked list.

- If the user selects option 5, print a goodbye message and exit the program.

- For any other input, print an error message and prompt the user to select a valid option.

8. Continue displaying the menu and prompting the user to select an option until the user enters 5 to quit the program.

9. Close the file and exit the program.

Qn1_function2.c

1. Include the necessary libraries and header files.
2. Define the main function sort which accepts an array of structs info and its length arrayLength.
3. Inside the sort function, prompt the user to choose a sorting technique and the preferred order (ascending or descending) of the sort.
4. Depending on the user's choice, call the corresponding sorting function.
5. Define the bubble sort in ascending order function BubbleSortAsc which accepts the same parameters as sort.
6. In the BubbleSortAsc function, loop through the unsorted array and compare the player name of the current element with the player name of the next element.
7. If the current element's player name comes after the next element's player name, swap the current element with the next element to move the bigger element to the right.
8. Define the bubble sort in descending order function BubbleSortDesc which accepts the same parameters as sort.
9. In the BubbleSortDesc function, loop through the unsorted array and compare the player name of the current element with the player name of the next element.
10. If the current element's player name comes before the next element's player name, swap the current element with the next element to move the bigger element to the left.
11. Define the selection sort in ascending order function SelectionSortAsc which accepts the same parameters as sort.
12. In the SelectionSortAsc function, loop through the unsorted array and find the minimum element.
13. Swap the minimum element with the first unsorted element.
14. Define the selection sort in descending order function SelectionSortDesc which accepts the same parameters as sort.
15. In the SelectionSortDesc function, loop through the unsorted array and find the maximum element.
16. Swap the maximum element with the first unsorted element.

Qn1_function3.c

1. Include the header files "Qn1_struct_function12.h" and "Qn1_function34.h".
2. Inside the searching function, prompt the user to enter the player name to search.
3. Prompt the user to select the searching method: linear or binary.
4. Read the user's input for searching method.
5. Use a switch statement to call either the linear or binary search function depending on the user's choice.

6. If the current index is greater than the array length, print "No result found" and return -1.
7. If the current player name matches the search key, print the player information and return the current index.
8. If the current player name does not match the search key, recursively call the linear_search function with i+1 as the new index.
9. Sort the array in ascending order using the BubbleSortAsc function.
10. If the low bound is greater than the high bound, print "No result found" and return -1.
11. Calculate the middle index of the array.
12. If the search key matches the middle player name, print the player information and return the middle index.
13. If the search key is less than the middle player name, recursively call the binary_search function with the left sub-array.
14. If the search key is greater than the middle player name, recursively call the binary_search function with the right sub-array.
15. In the main function, return -1 if no result is found.

Qn1_function4.c

1. Call the "BubbleSortAsc" function from "Qn1_function34.h" to sort the "info" array in ascending order.

2. Declare a pointer to the "Info" struct and set it to NULL. Also, declare a second pointer to "Info" struct and set it to NULL.

3. Use a "for" loop to iterate through the "info" array.

4. Inside the loop, allocate memory for a new node and populate its fields with data from the current element of the "info" array.

5. If the linked list is empty, set the start pointer to the new node. Otherwise, set the next pointer of the current node to the new node.

6. Set the current node pointer to the new node.

7. Print the header for the scoreboard.

8. Use a "while" loop to iterate through the linked list and print each node's data.

9. Set the start pointer to the next node in the list.

10. End the function.

4. Source code:

Qn1.c

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: Qn1.c
Brief explanation: the main program for Qn1, use for linking all of the functions
together.
Input: option
Output: Display the menu and 5 options.
Date created: 5/1/2023
Last modified: 5/9/2023
*/

//include libraries using the following
#include "Qn1_struct_function12.h"
#include "Qn1_function34.h"

// function 1: Display the details of the array
int displayFile(struct info info[], int arrayLength)
{
   printf("----------------------------------------------------------------------\n");
   printf("\t\t\tList of all player\n");
   printf("----------------------------------------------------------------------\n");
   printf("Player\tAge\tRank\tScore \n");

   // loop through the list and print each node
   for (int i = 0; i < SIZE; i++)
   {
      printf("%s\t%d\t%d\t%lf \n",
             info[i].player,
             info[i].age,
             info[i].rank,
             info[i].score);
   }
   return -1;
}

int main(int i, int low, int high)
{
            struct info info[SIZE];
   int option;

   //open file in reading mode
   FILE *fp = fopen("data.txt", "r");
   if (fp == NULL)
   {
      printf("Error opening the file \n");
```

```
    return -1;
}

for (int i = 0; i < SIZE; i++)
{
    fscanf(fp, "%s %d %d %lf",
    info[i].player,
    &info[i].age,
    &info[i].rank,
    &info[i].score);
}
// close the file
fclose(fp);

do
{
    //display the menu
    printf("---------------------------------------------------------------------\n");
    printf("\t\tWelcome to Amazing Player Scoreboard \n");
    printf("---------------------------------------------------------------------\n");
    printf("(1) Displaying all players \n");
    printf("(2) Sorting the data \n");
    printf("(3) Searching for a player \n");
    printf("(4) Inserting array to linked list \n");
    printf("(5) Quit Program \n");
    printf("Select your option: ");
    scanf("%d", &option); // add this line to read the selected option

    switch(option)
    {
        case 1:
        //call displayFile function
            displayFile(info,SIZE);
            break;

        case 2:
        //call sort function
            sort(info,SIZE);
        //call displayFile function after sorting
            displayFile(info,SIZE);
            break;

        case 3:
        //call search function to search for a player
            searching(info,i,low,high,SIZE);
                                break;

        case 4:
        //call insert function to insert array to linked list
            linked_list(info,SIZE);
            break;

          case 5:
          //print out the goodbye message
                              printf("---------------------------------------------------------------
--------\n");
            printf("Exiting the program...\n");
                                    printf("-------------------------------------------------------
--------------\n");
            break;

          //default case for other options that user type in
          default:
            printf("Invalid option selected. Please try again.\n");
            break;
        }
    } while (option != 5); //quit the program when user type in 5

    return 0;
}
```

Qn1_function2.c

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: Qn1_function2.c
Brief explanation: a separate c file for two sorting techniques- bubble and selection.
Input: sort_option, sort_order.
Output: the list in order which is chosen by the user.
Date created: 5/1/2023
Last modified: 5/9/2023
*/

//include libraries
#include "Qn1_struct_function12.h"
#include "Qn1_function34.h"

//function 2 main: sort by player name
void sort(struct info info[], int arrayLength)
{
          int sort_option;
          char sort_order[5];

          //print out to ask the user about the sorting technique
    printf("-------------------------------------------------------------------\n");
          printf("Select a sorting techniques: \n");
          printf("Press '1' for Bubble Sort \n");
```

```
        printf("Press '2' for Selection Sort \n");
  printf("Select your option: ");
        scanf("%d",&sort_option);

        //print out to ask the user about the sorting order
  printf("----------------------------------------------------------------------\n");
        printf("Select your preferred order by name: \n");
        printf("Type 'asc' for ascending order. \n");
        printf("Type 'desc' for descending order. \n");
  printf("Select your option: ");
        scanf("%s",sort_order);

        switch(sort_option)
        {
    case 1:
        if(strcmp(sort_order, "asc") == 0) //if user choose ascending order by using
bubble sort
            BubbleSortAsc(info,SIZE);

        else if(strcmp(sort_order, "desc") == 0) //if user choose descending order by
using bubble sort
            BubbleSortDesc(info,SIZE);
        break;

                case 2:
        if(strcmp(sort_order, "asc") == 0) //if user choose ascending order by using
selection sort
            SelectionSortAsc(info,SIZE);

        else if(strcmp(sort_order, "desc") == 0) //if user choose descending order by
using selection sort
            SelectionSortDesc(info,SIZE);
        break;

                default:
                        printf("Please type a valid option \n"); //if user choose an
invalid option
                break;
        }
}

//function 2 mini: bubble sort
//bubble sort in ascending order
void BubbleSortAsc(struct info info[], int arrayLength)
{
  // Open the file in write mode
  FILE* fp = fopen("data.txt", "w");
  if (fp == NULL)
  {
    printf("Error opening file to update.\n");
```

```
        return;
    }

    // Bubble sort algorithm
    for (int i = 0; i < (arrayLength - 1); ++i)
    {
        for (int j = 0; j < arrayLength - 1 - i; ++j)
        {
            if (strcmp(info[j].player, info[j+1].player) > 0)
            {
                struct info temp = info[j+1];
                info[j+1] = info[j];
                info[j] = temp;
            }
        }
    }
    // Write the sorted data to the file
    for (int i = 0; i < arrayLength; i++)
    {
        fprintf(fp, "%s %d %d %lf\n",
            info[i].player,
            info[i].age,
            info[i].rank,
            info[i].score);
    }
    // Close the file
    fclose(fp);
}


//bubble sort in descending order
void BubbleSortDesc(struct info info[], int arrayLength)
{
            int i, j;
            struct info temp;

    // Open the file in write mode
    FILE* fp = fopen("data.txt", "w");
    if (fp == NULL)
    {
        printf("Error opening file to update.\n");
        return;
    }

            for (i = 0; i < (arrayLength - 1); ++i)
            {
                        for (j = 0; j < arrayLength - 1 - i; ++j)
                        {
                                    // if the current element's player name comes before the next
element's player name
```

```
            // swap the current element with the next element to move the bigger element
to the left
                        if (strcmp(info[j].player, info[j+1].player) < 0)
                        {
                                temp = info[j+1];
                                info[j+1] = info[j];
                                info[j] = temp;
                        }
                }
            }
   // Write the sorted data to the file
   for (int i = 0; i < arrayLength; i++)
   {
      fprintf(fp, "%s %d %d %lf\n",
         info[i].player,
         info[i].age,
         info[i].rank,
         info[i].score);
   }
   // Close the file
   fclose(fp);
}

//function 2 mini: selection sort
//selection sort in ascending order
void SelectionSortAsc(struct info info[], int arrayLength)
{
   // Open the file in write mode
   FILE* fp = fopen("data.txt", "w");
   if (fp == NULL)
   {
      printf("Error opening file to update.\n");
      return;
   }

        for (int i = 0; i < arrayLength-1; ++i) // loop through the unsorted array
        {
                int j, min;
           struct info temp; // create temporary struct variable
                min = i;
                for (j = i+1; j < arrayLength; ++j) // loop through the remaining
unsorted array
                {
                        if (strcmp(info[j].player, info[min].player) < 0) // compare the
player name of the current element with the minimum element
                        min = j; // if the current element is smaller, update the
minimum index
                }
                temp = info[i]; // swap the minimum element with the first unsorted
element
```

```
            info[i] = info[min];
            info[min] = temp;
        }
    // Write the sorted data to the file
    for (int i = 0; i < arrayLength; i++)
    {
        fprintf(fp, "%s %d %d %lf\n",
            info[i].player,
            info[i].age,
            info[i].rank,
            info[i].score);
    }
    // Close the file
    fclose(fp);
}


//selection sort in descending order
void SelectionSortDesc(struct info info[], int arrayLength)
{
    int i;
    // Open the file in write mode
    FILE* fp = fopen("data.txt", "w");
    if (fp == NULL)
    {
        printf("Error opening file to update.\n");
        return;
    }
    // Loop through the array, starting from the first element
    for (i = 0; i < arrayLength-1; ++i)
    {
        int j, max;
        struct info temp;
        // Set the current element as the minimum
        max = i;
        // Find the minimum element in the rest of the array
        for (j = i+1; j < arrayLength; ++j)
        {
            if (strcmp(info[j].player, info[max].player) > 0)
            // If the current element is greater than the maximum, update the maximum
            max = j;
        }
        // Swap the maximum element with the current element
        temp = info[i];
        info[i] = info[max];
        info[max] = temp;
    }
    // Write the sorted data to the file
    for (int i = 0; i < arrayLength; i++)
    {
        fprintf(fp, "%s %d %d %lf\n",
            info[i].player,
            info[i].age,
            info[i].rank,
            info[i].score);
    }
    // Close the file
    fclose(fp);
}
```

Qn1_function3.c

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: Qn1_function3.c
Brief explanation: a separate c file for searching players in two ways, linear and
binary search.
Input: search_key, search_option.
Output: One specific player is found or not.
Date created: 5/1/2023
Last modified: 5/9/2023
*/

#include "Qn1_struct_function12.h"
#include "Qn1_function34.h"

//function 3 main: search by  string
//user can select linear or binary

int searching(struct info info[],int i,int low,int high, int arrayLength)
{
    printf("------------------------------------------------------------------------\n");
    printf("Enter the player name to search: ");
    char search_key[5];
    scanf("%s",search_key); //read the user's input for player name

    printf("------------------------------------------------------------------------\n");
    printf("Select the searching method: \n");
    printf("Press '1' for linear search \n");
    printf("Press '2' for binary search \n");
    printf("Select your option: "); //ask the user to select the searching method
    int search_option;
    scanf("%d",&search_option); //read the user's input for searching method

    switch(search_option)
    {
        case 1:
          linear_search(search_key,info,0,arrayLength); //if user choose linear search,
then call linear search function
          break;

       case 2:
          binary_search(search_key,info,0,arrayLength-1); //low=0 and
high=arrayLength-1
          break;
   }
   return -1;
}

//function 3 mini: linear search
int linear_search(const char *search_key, struct info info[],int i, int arrayLength)
{
   if (i > arrayLength) { //if the index is greater than array length, then no results are
returned
       printf("----------------------------------------------------------------------\n");
       printf("No result found. \n");
       return -1;
   }
   else if (strcmp(info[i].player, search_key) == 0)
   {
       printf("----------------------------------------------------------------------\n");
       printf("Result found: \n");
       printf("%s\t%d\t%d\t%lf \n",
       info[i].player,
       info[i].age,
       info[i].rank,
       info[i].score);
       return i;
   }
   else
   {
       return linear_search(search_key, info,i+1, arrayLength); //function called itself
(recursive)
   }
}


//function 3 mini: binary search
int binary_search(const char *search_key, struct info info[], int low, int high)
{
   BubbleSortAsc(info, SIZE);

   if (low > high)
   {
       printf("----------------------------------------------------------------------\n");
       printf("No result found. \n");
       return -1;
```

```
    }
    else
    {
        int middle = (low + high) / 2; //find the middle of the array
        if (strcmp(search_key, info[middle].player) == 0) //if the search key is found
        {
            //print out the result
            printf("----------------------------------------------------------------------\n");
            printf("Result found: \n");
            printf("%s\t%d\t%d\t%lf \n",
            info[middle].player,
            info[middle].age,
            info[middle].rank,
            info[middle].score);
            return middle;
        }
        else if (strcmp(search_key, info[middle].player) < 0) //if the search key is less
than the middle
        {
            return binary_search(search_key, info, low, middle - 1); //call the function to
search the left sub-array
        }
        else //if the search key is greater than the middle
        {
            return binary_search(search_key, info, middle + 1, high); //call the function to
search the right sub-array
        }
    }
}
```

Qn1_function4.c

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: Qn1_function4.c
Brief explanation: the separate C file for the linked list function, use to transfer the
data from the array to the linked list.
Input: None
Output: Data of all players by using the linked list.
Date created: 5/1/2023
Last modified: 5/9/2023
*/

#include "Qn1_struct_function12.h"
#include "Qn1_function34.h"

void linked_list(struct info info[],int arrayLength)
{
    // Bubble sort the array in ascending order
    BubbleSortAsc(info, arrayLength);

    // Define a pointer to the Info struct and declare them as NULL
    typedef struct info Info;
    typedef Info *InfoPtr;
    InfoPtr startPtr = NULL;
    InfoPtr currentPtr = NULL;

    // Iterate through the array and create a new node for each element
    for (int i = 0; i < arrayLength; i++)
    {
        // Allocate memory for a new node and populate its fields with data from the
array
        InfoPtr newPtr = (Info*)malloc(sizeof(Info));
        strcpy(newPtr->player, info[i].player);
        newPtr->age = info[i].age;
        newPtr->rank = info[i].rank;
        newPtr->score = info[i].score;
        newPtr->next = NULL;

        // If the linked list is empty, set the start pointer to the new node
        if (currentPtr == NULL)
        {
            startPtr = newPtr;
        }
        // Otherwise, set the next pointer of the current node to the new node
        else
        {
            currentPtr->next = newPtr;
        }

        // Set the current node pointer to the new node
        currentPtr = newPtr;
    }

    // Print the header for the scoreboard
    printf("--------------------------------------------------------------------\n");
    printf("\t\tWelcome to Amazing Player Scoreboard (linked list) \n");
    printf("--------------------------------------------------------------------\n");
    printf("Player\tAge\tRank\tScore \n");

    // Iterate through the linked list and print each node's data
    while (startPtr != NULL)
    {
        printf("%s\t%d\t%d\t%lf \n",
            startPtr->player,
            startPtr->age,
            startPtr->rank,
            startPtr->score);
        // Set the start pointer to the next node in the list
        startPtr = startPtr->next;
    }
}
```

Qn1_function34.h

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: Qn1_struct_function12.h
Brief explanation: include function prototypes for searching and and linked list
transfer.
Input: None
Output: None
Date created: 5/1/2023
Last modified: 5/9/2023
*/
//prototype for function 3
int linear_search(const char *search_key, struct info info[],int i,int arrayLength);
int binary_search(const char *search_key, struct info info[],int low, int high);
//one prototype function to call the whole 4 functions above
int searching(struct info info[],int i,int low,int high, int arrayLength);

//prototype for function 4
void linked_list(struct info info[],int arrayLength);
```

Qn1_struct_function12.h

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: Qn1_struct_function12.h
Brief explanation: include libraries, info structure, and function prototypes for
displayFile and Sorting Techniques.
Input: None
Output: None
Date created: 5/1/2023
Last modified: 5/9/2023
*/

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

#define SIZE 10
```

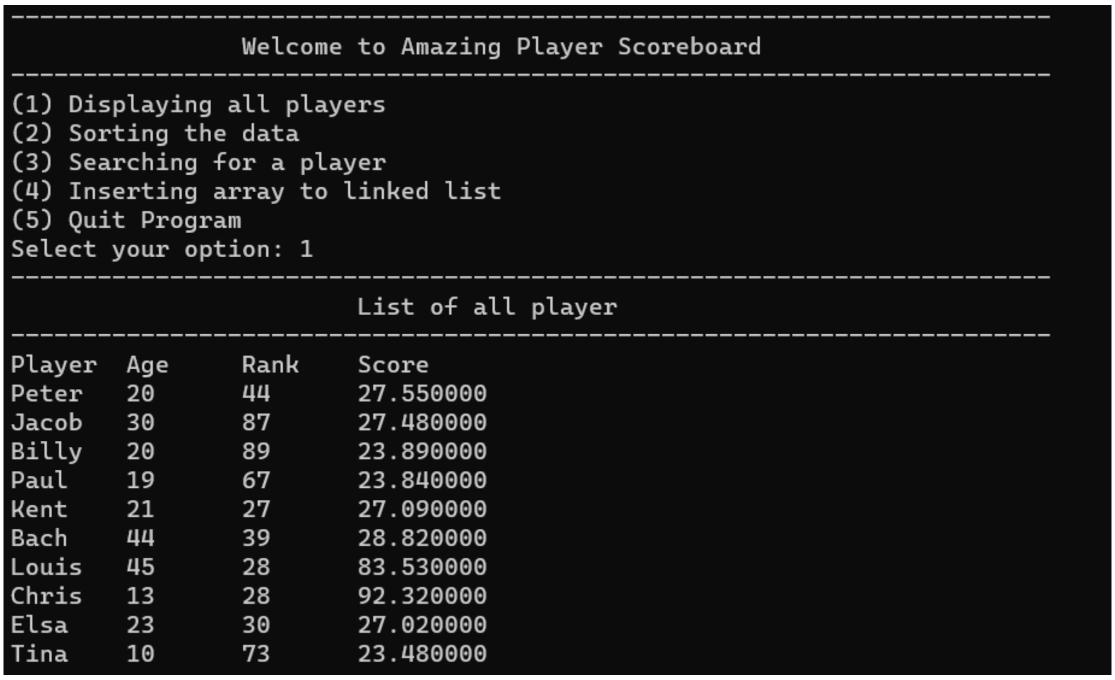
```
struct info
{
   char player[10];
   int age;
   int rank;
   double score;
   struct info *next;
};

//prototype for function 1
int displayFile(struct info info[], int arrayLength);

//prototype for function 2
void BubbleSortAsc(struct info info[], int arrayLength);
void BubbleSortDesc(struct info info[], int arrayLength);
void SelectionSortAsc(struct info info[], int arrayLength);
void SelectionSortDesc(struct info info[], int arrayLength);
void sort(struct info info[], int arrayLength);
```

data.txt

```
Peter   20      44      27.55
Jacob   30      87      27.48
Billy   20      89      23.89
Paul    19      67      23.84
Kent    21      27      27.09
Bach    44      39      28.82
Louis   45      28      83.53
Chris   13      28      92.32
Elsa    23      30      27.02
Tina    10      73      23.48
```

5. Screenshots showing working program (Show all possible outcome):

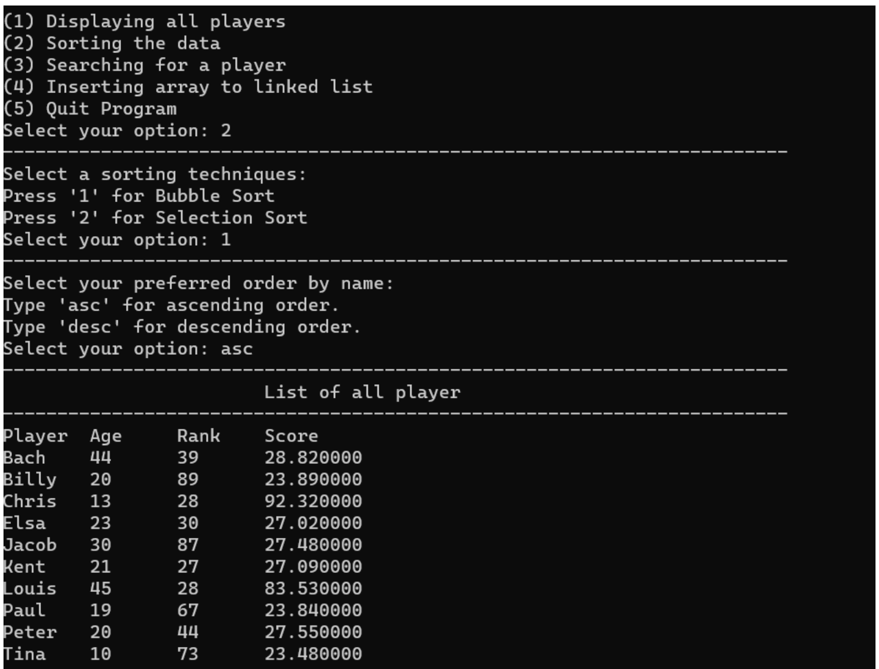
Option 1: display



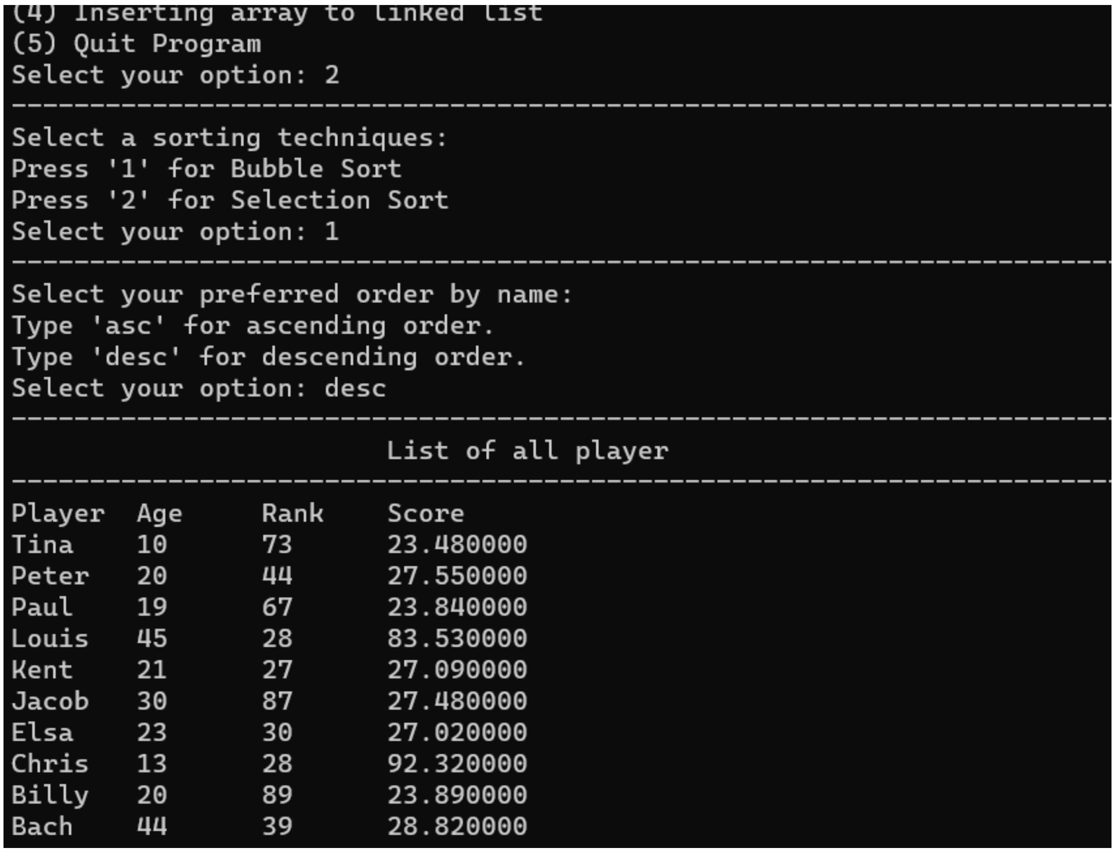
```
------------------------------------------------------------------------
                Welcome to Amazing Player Scoreboard
------------------------------------------------------------------------
(1) Displaying all players
(2) Sorting the data
(3) Searching for a player
(4) Inserting array to linked list
(5) Quit Program
Select your option: 1
------------------------------------------------------------------------
                        List of all player
------------------------------------------------------------------------
Player  Age     Rank    Score
Peter   20      44      27.550000
Jacob   30      87      27.480000
Billy   20      89      23.890000
Paul    19      67      23.840000
Kent    21      27      27.090000
Bach    44      39      28.820000
Louis   45      28      83.530000
Chris   13      28      92.320000
Elsa    23      30      27.020000
Tina    10      73      23.480000
```

Option 2: sorting by bubble ascending order

Option 2: sorting by bubble descending order

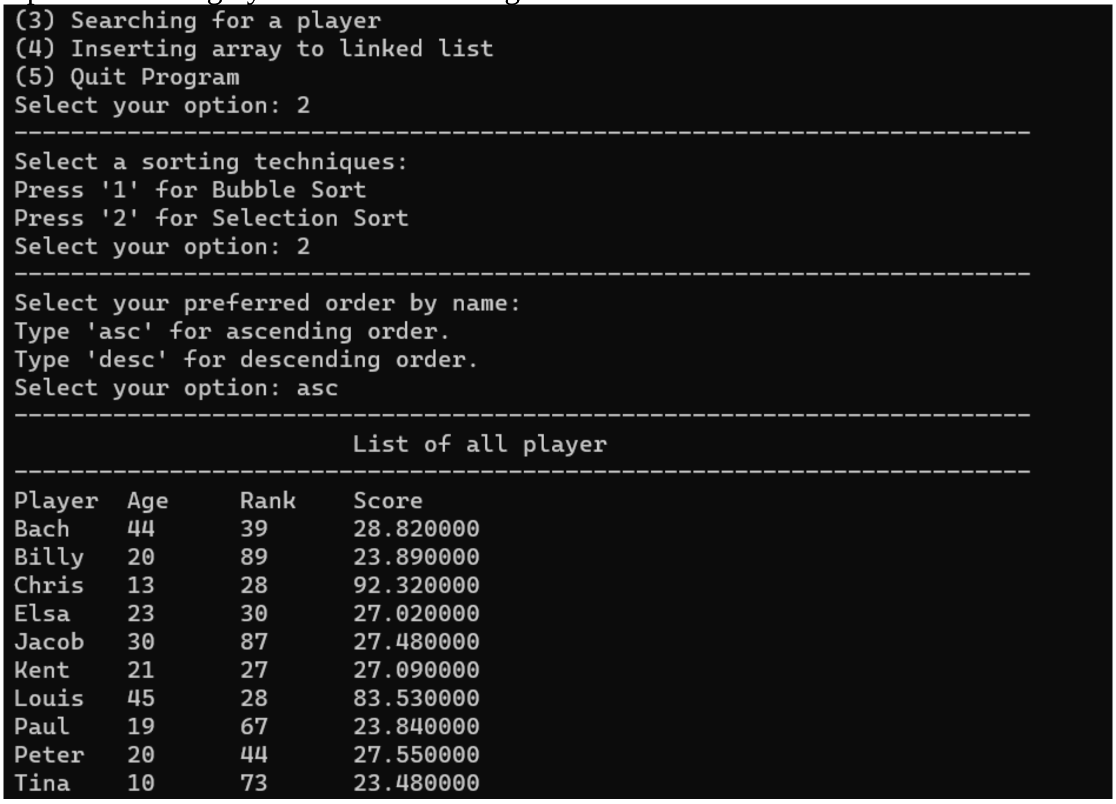


```
(1) Displaying all players
(2) Sorting the data
(3) Searching for a player
(4) Inserting array to linked list
(5) Quit Program
Select your option: 2
------------------------------------------------------------------------
Select a sorting techniques:
Press '1' for Bubble Sort
Press '2' for Selection Sort
Select your option: 1
------------------------------------------------------------------------
Select your preferred order by name:
Type 'asc' for ascending order.
Type 'desc' for descending order.
Select your option: asc
------------------------------------------------------------------------
                        List of all player
------------------------------------------------------------------------
Player  Age     Rank    Score
Bach    44      39      28.820000
Billy   20      89      23.890000
Chris   13      28      92.320000
Elsa    23      30      27.020000
Jacob   30      87      27.480000
Kent    21      27      27.090000
Louis   45      28      83.530000
Paul    19      67      23.840000
Peter   20      44      27.550000
Tina    10      73      23.480000
```



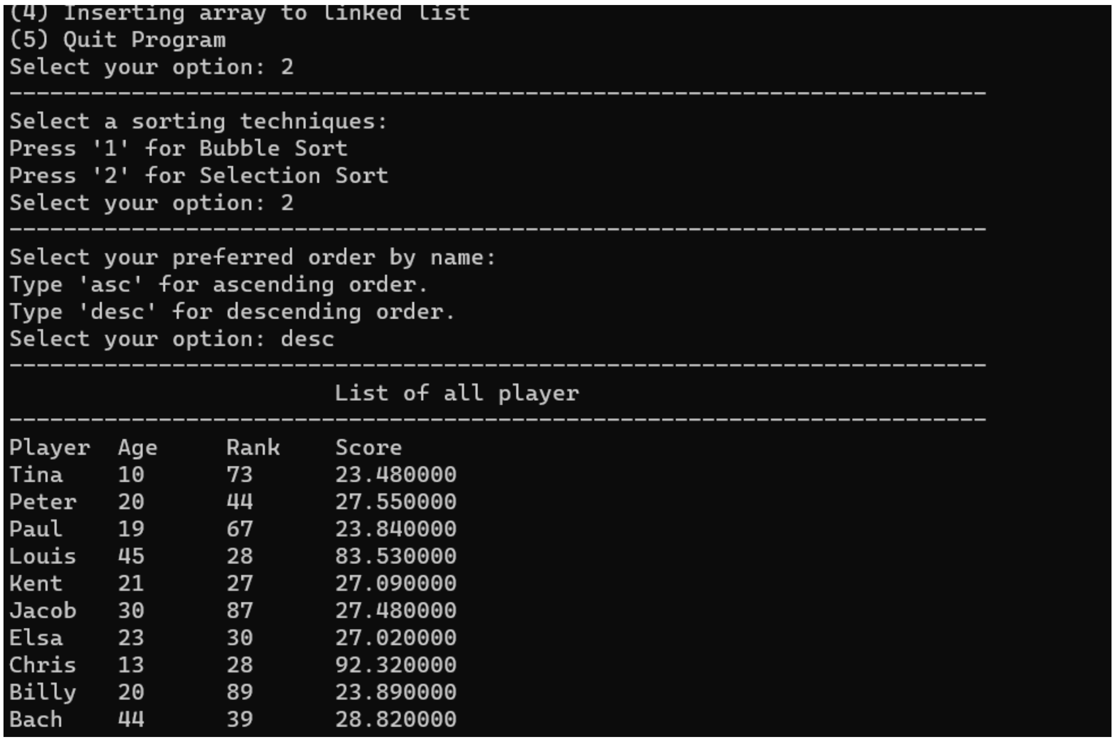
```
(4) Inserting array to linked list
(5) Quit Program
Select your option: 2
--------------------------------------------------------------------------------
Select a sorting techniques:
Press '1' for Bubble Sort
Press '2' for Selection Sort
Select your option: 1
--------------------------------------------------------------------------------
Select your preferred order by name:
Type 'asc' for ascending order.
Type 'desc' for descending order.
Select your option: desc
--------------------------------------------------------------------------------
                       List of all player
--------------------------------------------------------------------------------
Player  Age     Rank    Score
Tina    10      73      23.480000
Peter   20      44      27.550000
Paul    19      67      23.840000
Louis   45      28      83.530000
Kent    21      27      27.090000
Jacob   30      87      27.480000
Elsa    23      30      27.020000
Chris   13      28      92.320000
Billy   20      89      23.890000
Bach    44      39      28.820000
```

Option 2: sorting by selection ascending order



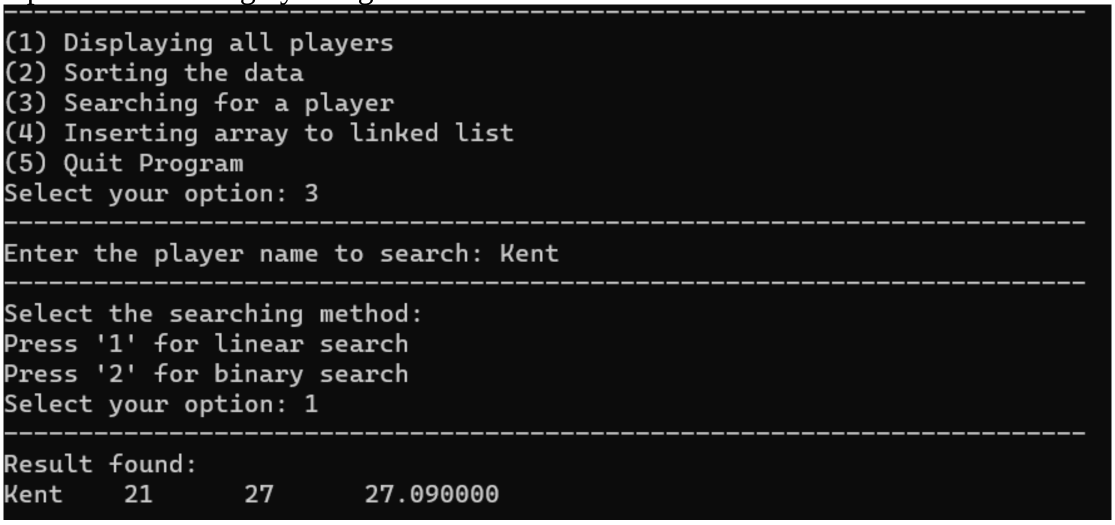
```
(3) Searching for a player
(4) Inserting array to linked list
(5) Quit Program
Select your option: 2
------------------------------------------------------------------------
Select a sorting techniques:
Press '1' for Bubble Sort
Press '2' for Selection Sort
Select your option: 2
------------------------------------------------------------------------
Select your preferred order by name:
Type 'asc' for ascending order.
Type 'desc' for descending order.
Select your option: asc
------------------------------------------------------------------------
                       List of all player
------------------------------------------------------------------------
Player  Age     Rank    Score
Bach    44      39      28.820000
Billy   20      89      23.890000
Chris   13      28      92.320000
Elsa    23      30      27.020000
Jacob   30      87      27.480000
Kent    21      27      27.090000
Louis   45      28      83.530000
Paul    19      67      23.840000
Peter   20      44      27.550000
Tina    10      73      23.480000
```

Option 2: sorting by selection descending order



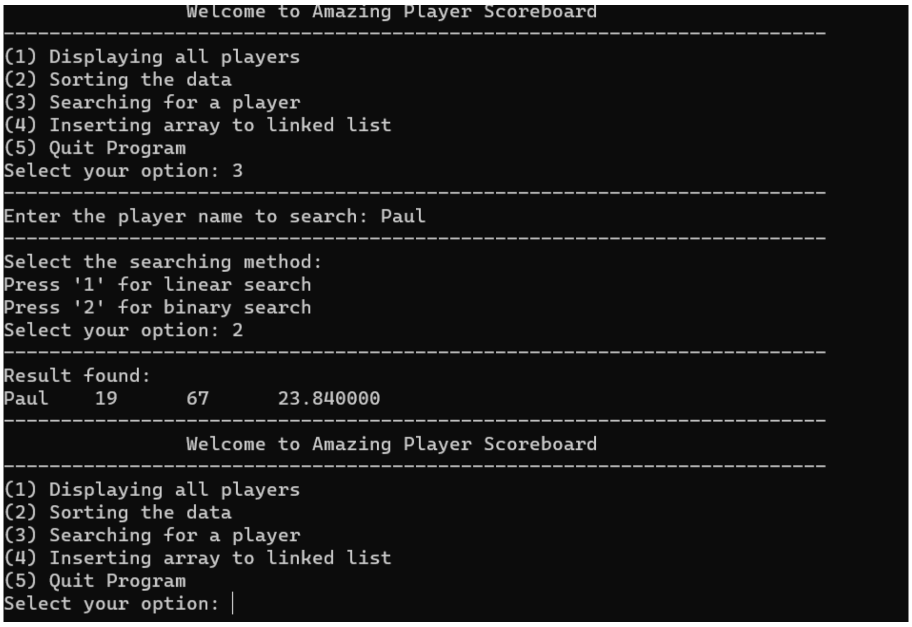
```
(4) Inserting array to linked list
(5) Quit Program
Select your option: 2
------------------------------------------------------------------------
Select a sorting techniques:
Press '1' for Bubble Sort
Press '2' for Selection Sort
Select your option: 2
------------------------------------------------------------------------
Select your preferred order by name:
Type 'asc' for ascending order.
Type 'desc' for descending order.
Select your option: desc
------------------------------------------------------------------------
                        List of all player
------------------------------------------------------------------------
Player  Age     Rank    Score
Tina    10      73      23.480000
Peter   20      44      27.550000
Paul    19      67      23.840000
Louis   45      28      83.530000
Kent    21      27      27.090000
Jacob   30      87      27.480000
Elsa    23      30      27.020000
Chris   13      28      92.320000
Billy   20      89      23.890000
Bach    44      39      28.820000
```

Option 3: searching by using linear search.



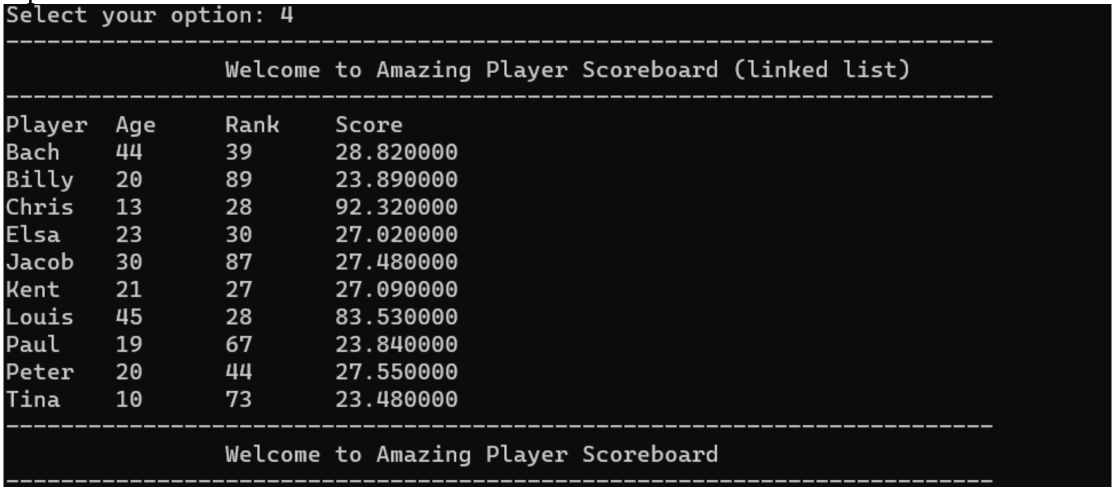
```
------------------------------------------------------------------------
(1) Displaying all players
(2) Sorting the data
(3) Searching for a player
(4) Inserting array to linked list
(5) Quit Program
Select your option: 3
------------------------------------------------------------------------
Enter the player name to search: Kent
------------------------------------------------------------------------
Select the searching method:
Press '1' for linear search
Press '2' for binary search
Select your option: 1
------------------------------------------------------------------------
Result found:
Kent    21      27      27.090000
```

Option 3: searching by using binary search



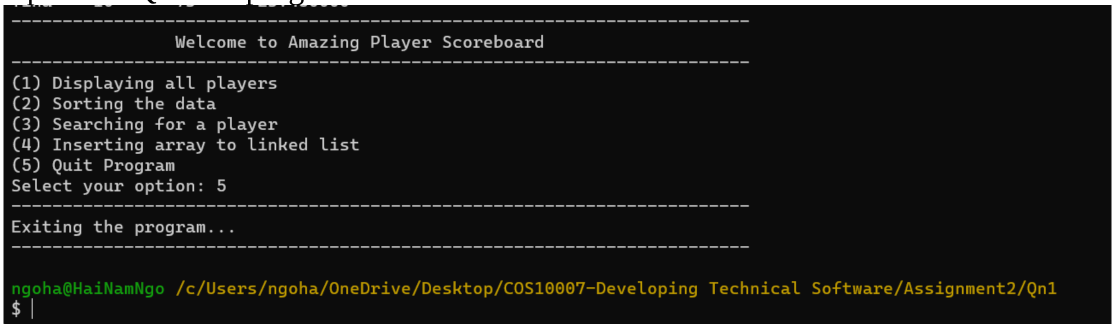
```
                Welcome to Amazing Player Scoreboard
------------------------------------------------------------------------
(1) Displaying all players
(2) Sorting the data
(3) Searching for a player
(4) Inserting array to linked list
(5) Quit Program
Select your option: 3
------------------------------------------------------------------------
Enter the player name to search: Paul
------------------------------------------------------------------------
Select the searching method:
Press '1' for linear search
Press '2' for binary search
Select your option: 2
------------------------------------------------------------------------
Result found:
Paul    19      67      23.840000
------------------------------------------------------------------------
                Welcome to Amazing Player Scoreboard
------------------------------------------------------------------------
(1) Displaying all players
(2) Sorting the data
(3) Searching for a player
(4) Inserting array to linked list
(5) Quit Program
Select your option: |
```

Option 4: Insert to linked list.



```
Select your option: 4
--------------------------------------------------------------------------
                Welcome to Amazing Player Scoreboard (linked list)
--------------------------------------------------------------------------
Player  Age     Rank    Score
Bach    44      39      28.820000
Billy   20      89      23.890000
Chris   13      28      92.320000
Elsa    23      30      27.020000
Jacob   30      87      27.480000
Kent    21      27      27.090000
Louis   45      28      83.530000
Paul    19      67      23.840000
Peter   20      44      27.550000
Tina    10      73      23.480000
--------------------------------------------------------------------------
                Welcome to Amazing Player Scoreboard
--------------------------------------------------------------------------
```

Option 5: Quit the program



```
------------------------------------------------------------------------
                Welcome to Amazing Player Scoreboard
------------------------------------------------------------------------
(1) Displaying all players
(2) Sorting the data
(3) Searching for a player
(4) Inserting array to linked list
(5) Quit Program
Select your option: 5
------------------------------------------------------------------------
Exiting the program...
------------------------------------------------------------------------

ngoha@HaiNamNgo /c/Users/ngoha/OneDrive/Desktop/COS10007-Developing Technical Software/Assignment2/Qn1
$ |
```

Question 2

1. Program description

This program checks the validity of a credit card number entered by the user. It prompts the user to input the credit card number as a series of digits and stores them in an array. Then, it calculates the checksum of the credit card number using a formula that involves finding the sums of certain digits in the number. If the checksum is divisible by 10, the credit card number is considered valid, otherwise, it is considered invalid. Finally, the program displays a message indicating whether the credit card number is valid or not.

2. Inputs and Outputs

The inputs and outputs for this program are described in Table 1.

Table 1. Data dictionary:

Data to be stored	**Sample data**	**Type of data**	**C type**	**Input method**	**In / Out**	**Variable name**
Digits in the credit card number	2	Number (1 digit)	Int array	cin	cout	cardNum

3. Algorithm

Program steps:

3. The program prompts the user to enter a credit card number as a series of digits into an array. The maximum number of digits is 20 and the user should use space between each digit and '-1' to end the series.

4. The program calculates the sum1 of the credit card number. Sum1 is the sum of every other digit in the credit card number, starting from every second digit and working backwards.

5. For each digit, the program doubles it and adds the digits of the result together. If the result is greater than or equal to 10, the sum of the two digits is used instead. The sum1 is printed to the console.

6. The program calculates the sum2 of the credit card number. Sum2 is the sum of every other digit in the credit card number, starting from the last digit and working backwards.

7. The sum2 is printed to the console.

8. The program calculates the check sum of the credit card number.

9. The check sum is calculated as the sum of sum1 and sum2 multiplied by 9.

10. The check sum is printed to the console.

11. The program checks if the check sum is divisible by 10.

12. If it is, the program prints a message indicating that the credit card number is valid. Otherwise, the program prints a message indicating that the credit card number is invalid.

13. The program ends.

1. Source code:

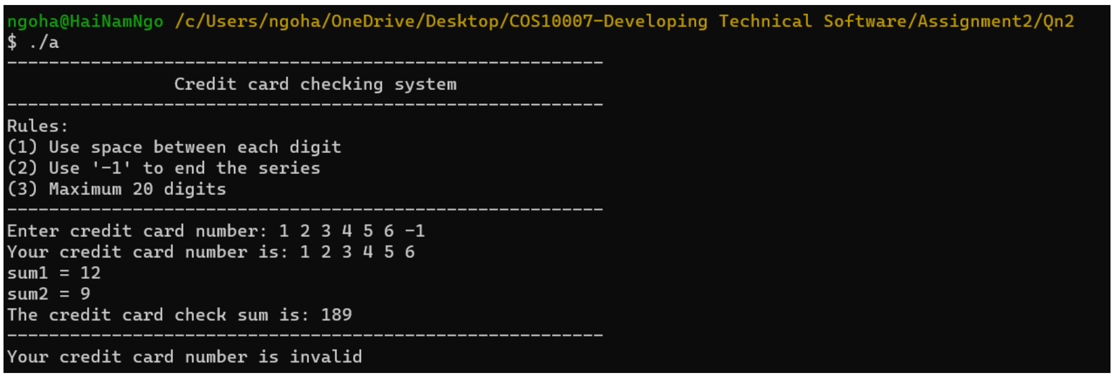
```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student id: 103488515
Name of the file: Qn2.cpp
Brief explanation: A C++ program that check if the credit card is valid or not by using
luhn algorithm.
Input: cardNum( series of number from the credit card)
Output: The system will check if it is valid or not.
Date created: 5/1/2023
Last modified: 5/13/2023
```

```
*/

#include <iostream> // allows program to perform input and output
#include <string>
#include <cmath>
using namespace std;

#define SIZE 20

//prototype of the function
int sum1(int cardNum[],int array_size);
int sum2(int cardNum[],int array_size);

int main()
{
   //Step1: Read in credit card number as a series of digits into an array
   int cardNum[SIZE];
   int i=0,number;
   //print out the menu
   cout <<"-------------------------------------------------------\n";
   cout <<"\t\tCredit card checking system\n";
   cout <<"-------------------------------------------------------\n";
   cout <<"Rules:\n(1) Use space between each digit\n(2) Use '-1' to end the
series\n(3) Maximum 20 digits\n";
   cout <<"-------------------------------------------------------\n";
   cout <<"Enter credit card number: ";

   //run the code when the user enters a valid number (below 20)
   while (i < SIZE)
   {
      cin >> number;
      if (number == -1)
      {
         break; // stop if user enters -1
      }
      cardNum[i] = number; //copy the number to the array
      i++; //increment the counter
   }

   // Print out the array
   cout << "Your credit card number is:";
   for (int j = 0; j < i; j++) {
      cout << " " << cardNum[j];
   }
   cout <<endl;

   //step 4: calculate check sum
   int checkSum = 0;

   //calculate the check sum
   checkSum = (sum1(cardNum,i) + sum2(cardNum,i)) * 9;
   cout << "The credit card check sum is: " << checkSum << endl;

   if (checkSum % 10 == 0) // if checkSum has the last digit equal to 0
   {
      cout <<"--------------------------------------------------------\n";
      cout << "Your credit card number is valid" << endl;
   }
   else
   {
      cout <<"--------------------------------------------------------\n";
      cout << "Your credit card number is invalid" << endl;
   }
   return 0;
}

//step 2: find sum1
int sum1(int cardNum[],int array_size)
{
   int j,doubled;
   int sum1=0;
   //start from the first second-digit, end continue to run to find other second-digits,
except the last digit in the series)
   for (j = 1; j < array_size-1;j+=2)
   {
      doubled = cardNum[j]; //copy the first digit to doubled
      doubled*=2; //multiply  by 2

      if (doubled >= 10) //if the number has two digits
      {
         sum1 += (doubled / 10) + (doubled % 10); //sum will be the sum of the two
digits
      }
      else
      {
         sum1+=doubled;
      }
   }

   cout << "sum1 = " << sum1 << endl;
   return sum1; //return the sum1
}

//step 3: find sum2
int sum2(int cardNum[],int array_size)
{
   int j;
   int sum2=0;

   for(j=0;j < array_size-1;j+=2)
```

```
    {
      sum2 += cardNum[j]; //sum will be the sum of every other digits except the last one
    }

    cout << "sum2 = " << sum2 << endl;
    return sum2;
}
```

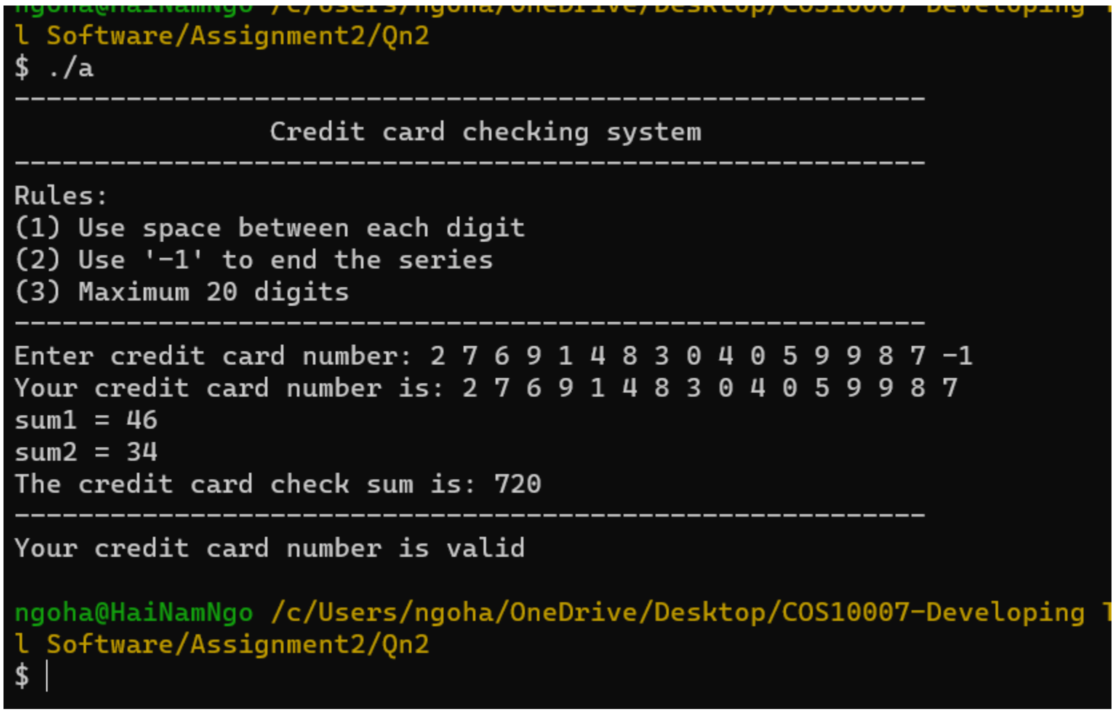
2. Screenshots showing working program (Show all possible outcome):

Is the credit card number is invalid.



```
ngoha@HaiNamNgo /c/Users/ngoha/OneDrive/Desktop/COS10007-Developing Technical Software/Assignment2/Qn2
$ ./a
---------------------------------------------------------
                Credit card checking system
---------------------------------------------------------
Rules:
(1) Use space between each digit
(2) Use '-1' to end the series
(3) Maximum 20 digits
---------------------------------------------------------
Enter credit card number: 1 2 3 4 5 6 -1
Your credit card number is: 1 2 3 4 5 6
sum1 = 12
sum2 = 9
The credit card check sum is: 189
---------------------------------------------------------
Your credit card number is invalid
```

If the credit card number is valid.



```
ngoha@HaiNamNgo /c/Users/ngoha/OneDrive/Desktop/COS10007-Developing T
l Software/Assignment2/Qn2
$ ./a
---------------------------------------------------------
                Credit card checking system
---------------------------------------------------------
Rules:
(1) Use space between each digit
(2) Use '-1' to end the series
(3) Maximum 20 digits
---------------------------------------------------------
Enter credit card number: 2 7 6 9 1 4 8 3 0 4 0 5 9 9 8 7 -1
Your credit card number is: 2 7 6 9 1 4 8 3 0 4 0 5 9 9 8 7
sum1 = 46
sum2 = 34
The credit card check sum is: 720
---------------------------------------------------------
Your credit card number is valid

ngoha@HaiNamNgo /c/Users/ngoha/OneDrive/Desktop/COS10007-Developing T
l Software/Assignment2/Qn2
$ |
```

Question 3

1. Program description

A C Program that deals with the purchase of televisions.

Overall, the program has 10 options, 9 of them are written by using functions.

The purchase order is processed on a first-come, first-served basis. When a consumer submits an order to purchase a television, the programme should search the database for that item, and if it is discovered, the purchase should be completed processed in accordance with the guidelines outlined here.

TVs in the shop are saved in a text file called TVs.txt.

The details of the TVs are read from TVs.txt and stored in a LinkedList at the start of the programme.

The list of TVs requested by clients is then saved in a queue. The programme then search for the order from the beginning of the queue in the LinkedList.

If the item is located in the list, delete it and place the TV in a stack so that it may be retrieved for the last TV sold.

2. Inputs and Outputs

The inputs and outputs for this program are described in Table 1.

Table 1. Data dictionary:

Data to be stored	**Sample data**	**Type of data**	**C type**	**Input method**	**In / Out**	**Variable name**
TV Id Code	104	number	Int	Fscanf scanf	Fprintf printf	Id
TV Name	“Apple TV”	Text: maximum 20 letters	String	Fscanf scanf	Fprintf printf	TVname
Menu option	2	number	int	scanf	functions	option

3. Algorithm

Program steps:

3. Read TV data from the file and store it in a linked list:
    - Define the function readTVs() that reads TV data from the file "TVs.txt" and returns a pointer to the start of the linked list.
    - Open the file using fopen().

- Initialize variables for the start pointer (startPtr), current pointer (currentPtr), and new pointer (newPtr).
- Read TV data from the file using fscanf() and dynamically allocate memory for each new TV node.
- Insert the new TV node into the linked list in alphabetical order based on the TV name.
- Close the file using fclose().

4. Add a new TV to the stock:
    - Define the function addTV() that takes a pointer to the start pointer (startPtr) as a parameter.
    - Allocate memory for a new TV node (newPtr).
    - Prompt the user to enter the ID and TV name for the new TV.
    - Traverse the linked list to find the correct position to insert the new TV in alphabetical order based on the TV name.
    - Update the pointers to insert the new TV node at the correct position.
    - If the new TV is the first in the list, update the start pointer.
    - Print a success message.
    - Display next order information:

    - Define the function queue_nextOrder_check() that takes the head and tail pointers of the order queue as parameters (headPtr and tailPtr).
    - Check if the head pointer is NULL, indicating no orders in the queue.
    - If the head pointer is not NULL, print the information of the next order (ID and TV name).
5. Display all orders:

    - Define the function displayOrder() that takes the head and tail pointers of the order queue as parameters (headPtr and tailPtr).
    - Check if the head pointer is NULL, indicating no orders in the queue.
    - If the head pointer is not NULL, traverse the order queue and print the information of each order (ID and TV name).
    - Add order to the queue:
    - Define the function add_order() that takes the start pointer of the TV stock, and the head and tail pointers of the order queue as parameters (startPtr, headPtr, and tailPtr).
    - Allocate memory for a new order node.
    - Prompt the user to enter the ID and TV name for the new order.
    - Traverse the TV stock linked list to check if the requested TV is available.
    - If the TV is available, add the new order to the queue by updating the head and tail pointers.
    - Print a success message if the order is added, or a message if the requested TV is not available.
6. Process the next order:

    - Define the function process_order() that takes pointers to the newest order stack, the start pointer of the TV stock, and the head and tail pointers of the order queue as parameters (newest, startPtr, headPtr, and tailPtr).
    - Check if the start pointer is NULL, indicating no orders in the queue.

- If the start pointer is not NULL, remove the next order from the queue and update the head pointer.
- Traverse the TV stock linked list to find the TV corresponding to the order.
- Create a new node in the newest order stack with the TV information.
- Remove the TV from the linked list.
- Print a success message.

7. Cancel last order:
   - Define the function cancel_order() that takes pointers to the newest
   - order stack, the start pointer of the TV stock, and the head and tail pointers of the order queue as parameters (newest, startPtr, headPtr, and tailPtr).

   - Check if the newest order stack is empty, indicating no orders to cancel.
   - If the newest order stack is not empty, remove the top order from the stack and update the newest order pointer.
   - Traverse the TV stock linked list to find the TV corresponding to the canceled order.
   - Insert the TV back into the linked list in alphabetical order based on the TV name.
   - Print a success message.
   8. Save TV stock to file:

   - Define the function saveTVs() that takes the start pointer of the TV stock as a parameter (startPtr).
   - Open the file "TVs.txt" using fopen() in write mode.
   - Traverse the TV stock linked list and write the TV data to the file using fprintf().
   - Close the file using fclose().

4. Source code:

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file:
Brief explanation: Create a program that manage the stock system and order system of a TV Store which has 10 options in the menu.
Input: option, id, TVname
Output: functions
Date created: 5/3/2023
Last modified: 5/15/2023
*/

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

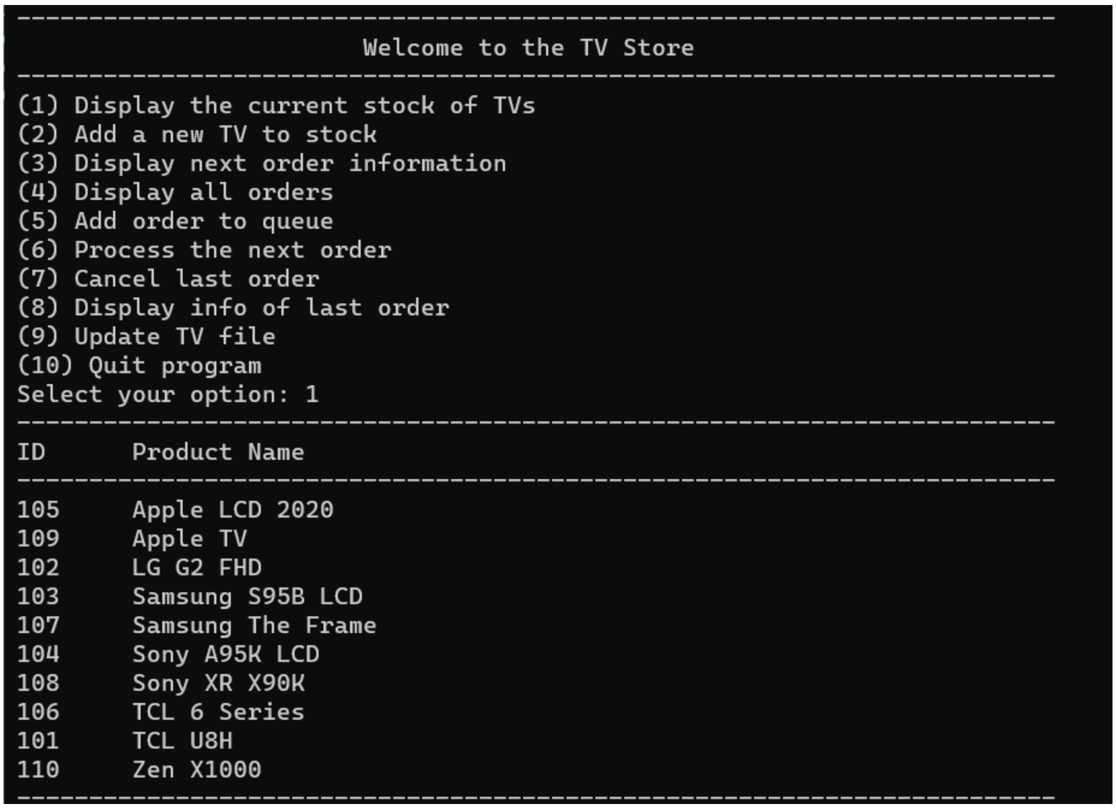
```
struct product
{
   int id;
   char TVname[20];
   struct product *next;
};

typedef struct product Product;
typedef Product *ProductPtr;

typedef struct product Order;
typedef Order *OrderPtr;

typedef struct product Stack;
typedef Stack *StackPtr;

//store the TVs.txt file in the linked list
ProductPtr readTVs()
{
   FILE *fp=fopen("TVs.txt","r");
   if(fp==NULL)
   {
      printf("Error opening the file \n");
      return NULL;
   }

   ProductPtr startPtr=NULL;
   ProductPtr currentPtr=NULL;
   ProductPtr newPtr=NULL;

   // Read the remaining TVs from the file
   while (!feof(fp))
   {
      newPtr = (Product*)malloc(sizeof(Product));
      fscanf(fp,"%d",&newPtr->id);
      fscanf(fp," %[^\n]s",newPtr->TVname);
      newPtr->next = NULL;

      if (startPtr == NULL || strcmp(newPtr->TVname, startPtr->TVname) < 0)
      {
         // If the list is empty or the new TV's name comes before the first TV's name
in the list,
         // insert the new TV at the beginning of the list
         newPtr->next = startPtr;
         startPtr = newPtr;
      }
      else
      {
         // Otherwise, find the position in the list where the new TV should be inserted
```

```
            currentPtr = startPtr;
            while (currentPtr->next != NULL && strcmp(newPtr->TVname, currentPtr-
>next->TVname) > 0)
            {
                // Traverse the list until we find the correct position to insert the new TV
                currentPtr = currentPtr->next;
            }
            // Insert the new TV into the list at the correct position
            newPtr->next = currentPtr->next;
            currentPtr->next = newPtr;
        }
    }
    // Close the file
    fclose(fp);
    return startPtr;
}

void addTV(ProductPtr *startPtr)
{
    ProductPtr newPtr, currentPtr, previousPtr;

    // allocate memory for new product
    newPtr = (Product*)malloc(sizeof(Product));

    printf("----------------------------------------------------------------------\n");
    printf("Enter a new TV details:\n");
    printf("ID: ");
    scanf("%d", &newPtr->id);
    getchar();
    printf("Product Name: ");
    scanf(" %[^\n]s",newPtr->TVname);
    newPtr->next = NULL;

    currentPtr = *startPtr;
    previousPtr = NULL;

    // find the correct position to insert new product in alphabetical order
    while (currentPtr != NULL)
    {
        if(strcmp(newPtr->TVname, currentPtr->TVname) > 0)
        {
            previousPtr = currentPtr;
            currentPtr = currentPtr->next;
        }
        // otherwise, insert the new product between previous and current
        else if (strcmp(newPtr->TVname, currentPtr->TVname) <= 0)
        {
            newPtr->next = currentPtr;
            previousPtr->next = newPtr;
            break;
        }
    }
    // if the new product is the first in the list, update the start pointer
    if (previousPtr == NULL)
    {
        newPtr->next = *startPtr;
        *startPtr = newPtr;
    }
    printf("----------------------------------------------------------------------\n");
    printf("The new TV has been added successfully!\n");
}



void queue_nextOrder_check(OrderPtr headPtr, OrderPtr tailPtr)
{
    if (headPtr == NULL)
    {
        printf("----------------------------------------------------------------------\n");
        printf("There is no order at the moment.\n");
    }
    else
    {
        printf("----------------------------------------------------------------------\n");
        printf("Next order information is:\n");

        printf("%d\t%s\n",
        headPtr->id,
        headPtr->TVname);
    }
}


void displayOrder(OrderPtr headPtr, OrderPtr tailPtr)
{
    if (headPtr == NULL)
    {
        printf("----------------------------------------------------------------------\n");
        printf("There is no order at the moment.\n");
    }
    else
    {
        printf("----------------------------------------------------------------------\n");
        printf("Order information is:\n");
        OrderPtr current = headPtr;
        while (current != NULL)
        {
            printf("%d\t%s\n",
            current->id,
            current->TVname);
            current = current->next;
        }
    }
}

void add_order(ProductPtr startPtr, OrderPtr *headPtr, OrderPtr *tailPtr)
{
    OrderPtr newOrder = (Order*)malloc(sizeof(Order));

    printf("----------------------------------------------------------------------\n");
    printf("Enter new order details:\n");
    printf("ID: ");
    scanf("%d", &newOrder->id);
    printf("Product Name: ");
    scanf(" %[^\n]s", newOrder->TVname);

    newOrder->next = NULL;

    ProductPtr currentPtr = startPtr;
    while (currentPtr != NULL)
    {
        if ((strcmp(newOrder->TVname, currentPtr->TVname) == 0)&&(currentPtr->id
== newOrder->id))
        {
            if (*headPtr == NULL)
            {
                *headPtr = newOrder;
                *tailPtr = newOrder;
            }
            else
            {
                (*tailPtr)->next = newOrder;
                *tailPtr = newOrder;
            }
            printf("----------------------------------------------------------------------\n");
            printf("The new order has been added successfully!\n");
            return;
        }
        currentPtr = currentPtr->next;
    }
    printf("----------------------------------------------------------------------\n");
    printf("The product is not in stock.\n");
}

//function 6:
void process_order(StackPtr *newest, ProductPtr *startPtr, OrderPtr *headPtr,
OrderPtr *tailPtr)
{
    // Move the head pointer to the next order
    OrderPtr tempPtr = *headPtr;
    *headPtr = (*headPtr)->next;

    if (*startPtr == NULL)
    {
        printf("----------------------------------------------------------------------\n");
        printf("There is no order to process at the moment.\n");
        return;
    }
    // If the queue is empty
    if (*headPtr == NULL)
    {
        *tailPtr = NULL;
    }

    ProductPtr currentPtr = *startPtr;
    StackPtr lastOrder = (Stack*)malloc(sizeof(Stack));
    lastOrder->next = *newest;
    *newest = lastOrder;

    while (currentPtr != NULL)
    {
        if ((strcmp(currentPtr->TVname, tempPtr->TVname) == 0) && (tempPtr->id ==
currentPtr->id))
        {
            lastOrder->id = currentPtr->id;
            strcpy(lastOrder->TVname, currentPtr->TVname);
            break;
        }
        currentPtr = currentPtr->next;
    }

    // Remove the TV from the linked list
    if (currentPtr == *startPtr)
    {
        // If the current TV node is the first node, update the startPtr to point to the next
node
        *startPtr = (*startPtr)->next;
    }
    else
    {
        // If the current TV node is not the first node,
        // traverse the linked list to find the previous node

        // Start with the first node as the previous node
        ProductPtr previousPtr = *startPtr;

        // Traverse the linked list until the next node of the previous node is the current
node
        while (previousPtr->next != currentPtr)
        {
            previousPtr = previousPtr->next;
        }
        // Update the next pointer of the previous node to skip over the current node
        previousPtr->next = currentPtr->next;
    }
    free(tempPtr);
    printf("----------------------------------------------------------------------\n");
    printf("The next order has been proceed.\n");
}

//function 7:
void cancel_order(StackPtr *newest, ProductPtr *startPtr)
{
    StackPtr lastOrder = *newest;

    ProductPtr currentPtr = *startPtr;
    ProductPtr previousPtr = NULL;

    // Find the correct position to insert the new product in alphabetical order
    while (currentPtr != NULL)
    {
        if (strcmp(lastOrder->TVname, currentPtr->TVname) > 0)
        {
            previousPtr = currentPtr;
            currentPtr = currentPtr->next;
        }
        else if (strcmp(lastOrder->TVname, currentPtr->TVname) <= 0)
        {
            // Insert the new product between previous and current
            lastOrder->next = currentPtr;
            if (previousPtr != NULL)
                previousPtr->next = lastOrder;
            else
                *startPtr = lastOrder; // New product is the first in the list
            break;
        }
    }

    // If the new product is the last in the list, update the *newest pointer
    if (currentPtr == NULL)
    {
        lastOrder->next = NULL;
        *newest = lastOrder;
    }
    printf("----------------------------------------------------------------------\n");
    printf("The last order has been canceled.\n");
}

void display_Order(StackPtr *newest)
{
    if (*newest == NULL)
    {
        printf("--------------------------------------------------------------------\n");
        printf("There is no order to process at the moment.\n");
    }
    else
    {
        printf("--------------------------------------------------------------------\n");
        printf("Information of last order is:\n");
        printf("%d\t%s\n",
        (*newest)->id,
        (*newest)->TVname);

    }
}

void updateFile(ProductPtr startPtr)
{
    // Open the file in append mode
    FILE* fp = fopen("TVs.txt", "w");
    if (fp == NULL)
    {
        printf("Error opening file to update.\n");
        return;
    }

    ProductPtr currentPtr = startPtr;
    while (currentPtr != NULL)
    {
        fprintf(fp, "%d\t%s\n", currentPtr->id, currentPtr->TVname);
        currentPtr = currentPtr->next;
    }

    // Close the file
    fclose(fp);

    printf("--------------------------------------------------------------------\n");
    printf("Update File Completed.\n");
}



//this function 1: is used to display TV list to the screen
void displayFile(ProductPtr startPtr)
{
    //print out the TV list
    printf("--------------------------------------------------------------------\n");
    printf("ID\tProduct Name \n");
    printf("--------------------------------------------------------------------\n");
        // loop through the list and print each node
        while (startPtr != NULL)
        {
                printf("%d\t%s \n",
                startPtr->id,
                startPtr->TVname);

                startPtr = startPtr->next;
        }
}

//this function is created to display the instruction to the user
int menu()
{
   int option;
   //display the menu
        printf("-----------------------------------------------------------------------\n");
        printf("\t\t\tWelcome to the TV Store\n");
   printf("-----------------------------------------------------------------------\n");
   printf("(1) Display the current stock of TVs \n");
   printf("(2) Add a new TV to stock \n");
   printf("(3) Display next order information \n");
   printf("(4) Display all orders \n");
   printf("(5) Add order to queue \n");
   printf("(6) Process the next order \n");
   printf("(7) Cancel last order \n");
   printf("(8) Display info of last order \n");
   printf("(9) Update TV file \n");
   printf("(10) Quit program \n");
   printf("Select your option: ");
   scanf("%d",&option);

   return option; // add this line to return the selected option to the main function
}

//main function to link all of the functions together
void main()
{
   ProductPtr startPtr = readTVs();
   ProductPtr headPtr = NULL;
   ProductPtr tailPtr = NULL;
   StackPtr newest= NULL;

   int option;
   do
   {
      option = menu();
      if (option == 1)
      {
         displayFile(startPtr);
      }
```

```
        else if (option == 2)
        {
            addTV(&startPtr);
        }
        else if (option == 3)
        {
            queue_nextOrder_check(headPtr, tailPtr);
        }
        else if (option == 4)
        {
            displayOrder(headPtr, tailPtr);
        }
        else if (option == 5)
        {
            add_order(startPtr, &headPtr, &tailPtr);
        }
        else if (option == 6)
        {
            process_order(&newest, &startPtr, &headPtr, &tailPtr);
        }
        else if (option == 7)
        {
            cancel_order(&newest, &startPtr);
        }
        else if (option == 8)
        {
            display_Order(&newest);
        }

        else if (option == 9)
        {
            updateFile(startPtr);
        }

        else if (option == 10)
        {
            printf("----------------------------------------------------------------------\n");
            printf("Exiting the program...\n");
                        printf("-------------------------------------------------------------------
----\n");
        }
    } while (option != 10);
}
```

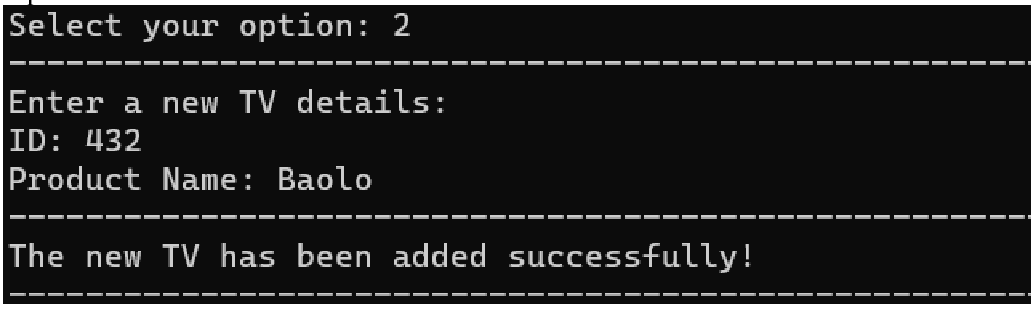
5. Screenshots showing working program (Show all possible outcome):

Menu and option 1:



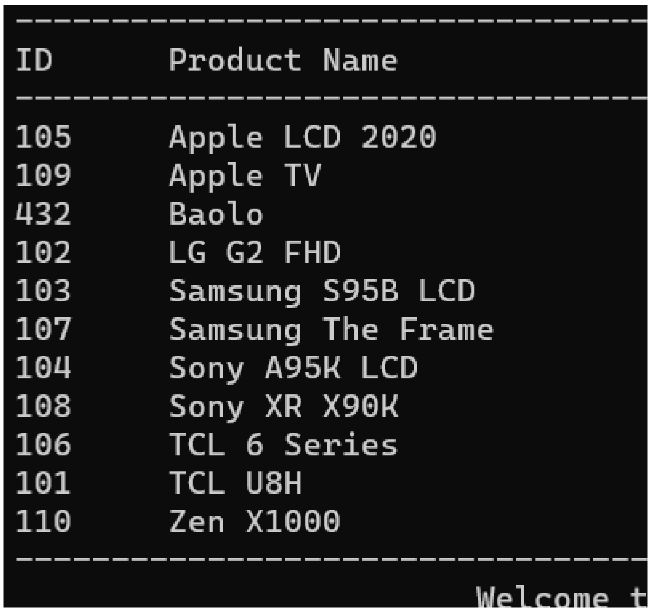
```
------------------------------------------------------------------------
                        Welcome to the TV Store
------------------------------------------------------------------------
(1) Display the current stock of TVs
(2) Add a new TV to stock
(3) Display next order information
(4) Display all orders
(5) Add order to queue
(6) Process the next order
(7) Cancel last order
(8) Display info of last order
(9) Update TV file
(10) Quit program
Select your option: 1
------------------------------------------------------------------------
ID      Product Name
------------------------------------------------------------------------
105     Apple LCD 2020
109     Apple TV
102     LG G2 FHD
103     Samsung S95B LCD
107     Samsung The Frame
104     Sony A95K LCD
108     Sony XR X90K
106     TCL 6 Series
101     TCL U8H
110     Zen X1000
------------------------------------------------------------------------
```

Option 2:



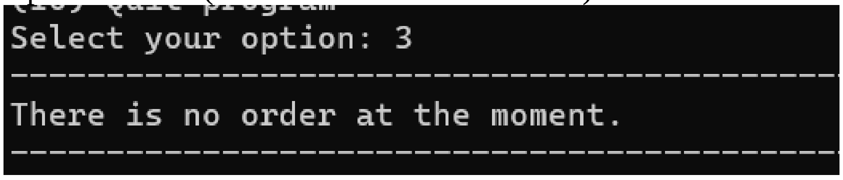
```
Select your option: 2
----------------------------------------------------
Enter a new TV details:
ID: 432
Product Name: Baolo
----------------------------------------------------
The new TV has been added successfully!
----------------------------------------------------
```

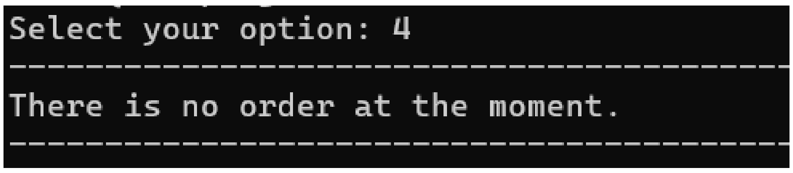
After added:



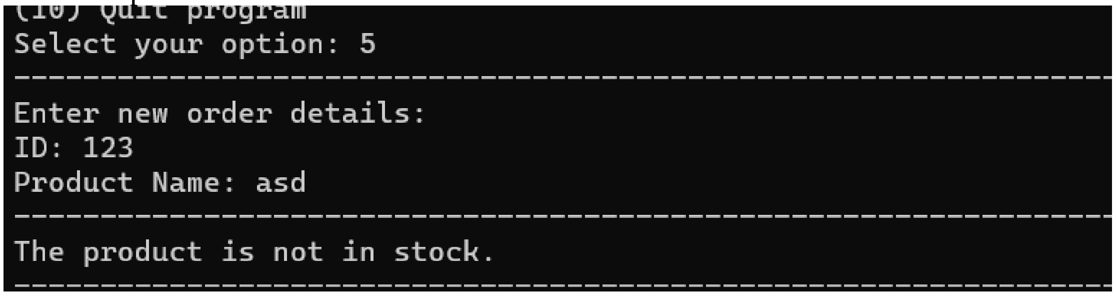
```
---------------------------------
ID      Product Name
---------------------------------
105     Apple LCD 2020
109     Apple TV
432     Baolo
102     LG G2 FHD
103     Samsung S95B LCD
107     Samsung The Frame
104     Sony A95K LCD
108     Sony XR X90K
106     TCL 6 Series
101     TCL U8H
110     Zen X1000
---------------------------------
                        Welcome t
```

Option 3 and 4 (No order at the moment case)

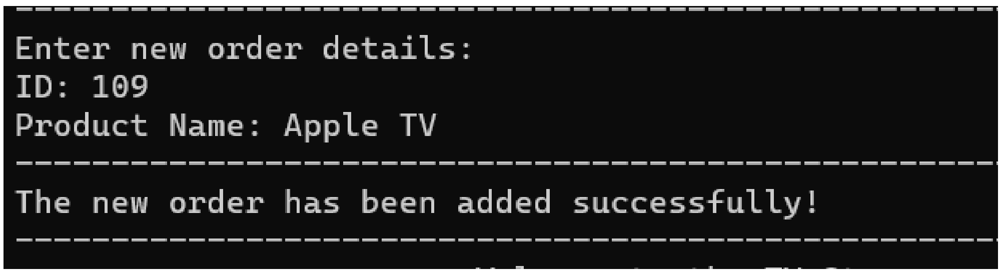

```
(10) Quit program
Select your option: 3
--------------------------------------------
There is no order at the moment.
--------------------------------------------
```

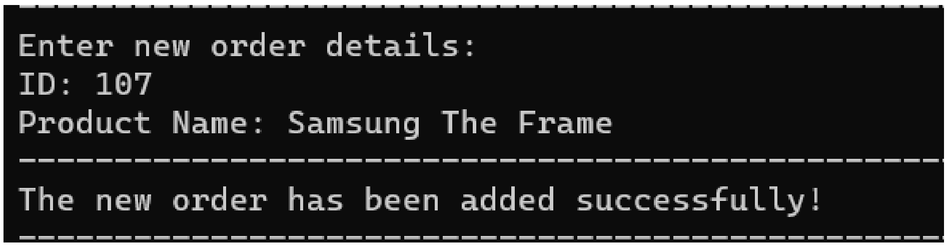

```
Select your option: 4
-----------------------------------------
There is no order at the moment.
-----------------------------------------
```

Option 5:
Case: No product found in the stock.

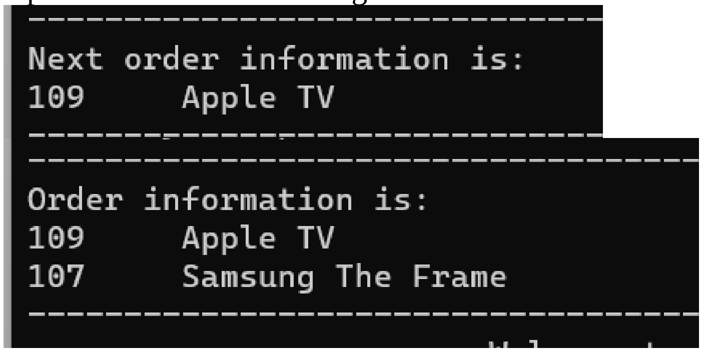

```
(10) Quit program
Select your option: 5
----------------------------------------------------------------
Enter new order details:
ID: 123
Product Name: asd
----------------------------------------------------------------
The product is not in stock.
----------------------------------------------------------------
```

Case: Product found in the stock.

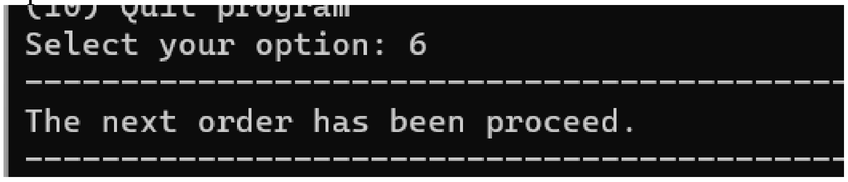

```
--------------------------------------------------
Enter new order details:
ID: 109
Product Name: Apple TV
--------------------------------------------------
The new order has been added successfully!
--------------------------------------------------
```


```
-----------------------------------------------
Enter new order details:
ID: 107
Product Name: Samsung The Frame
-----------------------------------------------
The new order has been added successfully!
-----------------------------------------------
```

Option 3 and 4 when having order.

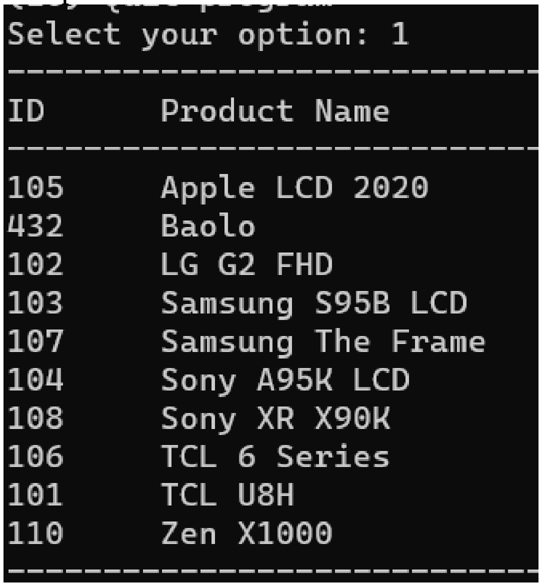

```
--------------------------------
Next order information is:
109      Apple TV
--------------------------------
-------------------------------------
Order information is:
109      Apple TV
107      Samsung The Frame
-------------------------------------
```

Option 6:



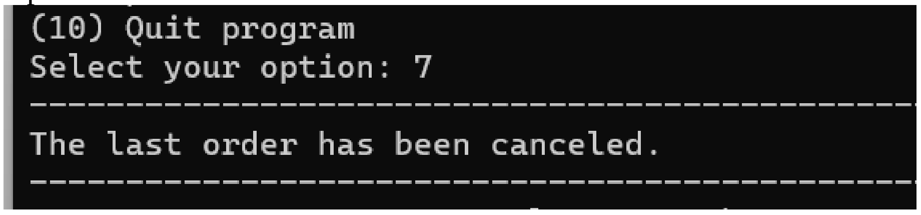
```
(10) Quit program
Select your option: 6
-------------------------------------------
The next order has been proceed.
-------------------------------------------
```

The product is removed from the stock.



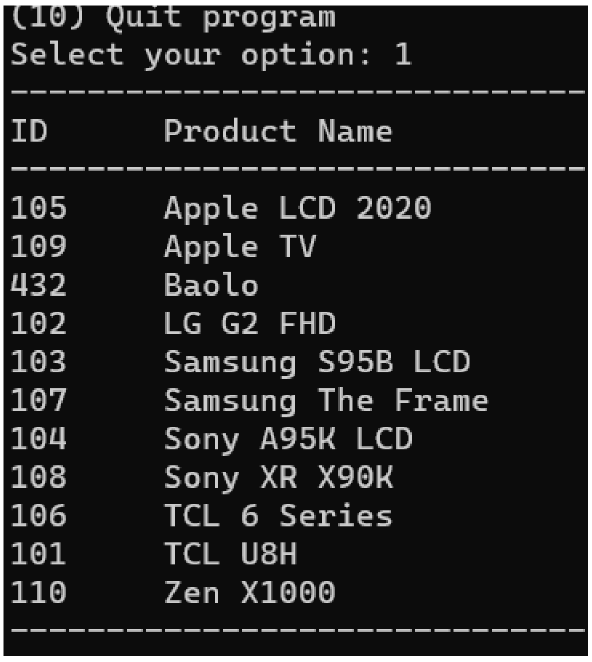
```
Select your option: 1
----------------------------
ID      Product Name
----------------------------
105     Apple LCD 2020
432     Baolo
102     LG G2 FHD
103     Samsung S95B LCD
107     Samsung The Frame
104     Sony A95K LCD
108     Sony XR X90K
106     TCL 6 Series
101     TCL U8H
110     Zen X1000
----------------------------
```

Option 7:



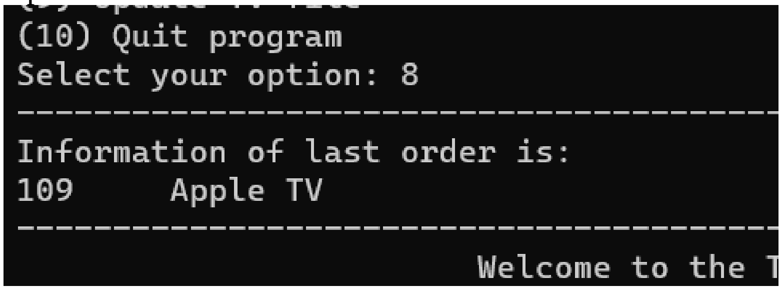
```
(10) Quit program
Select your option: 7
-----------------------------------------------
The last order has been canceled.
-----------------------------------------------
```

The item re-stock.



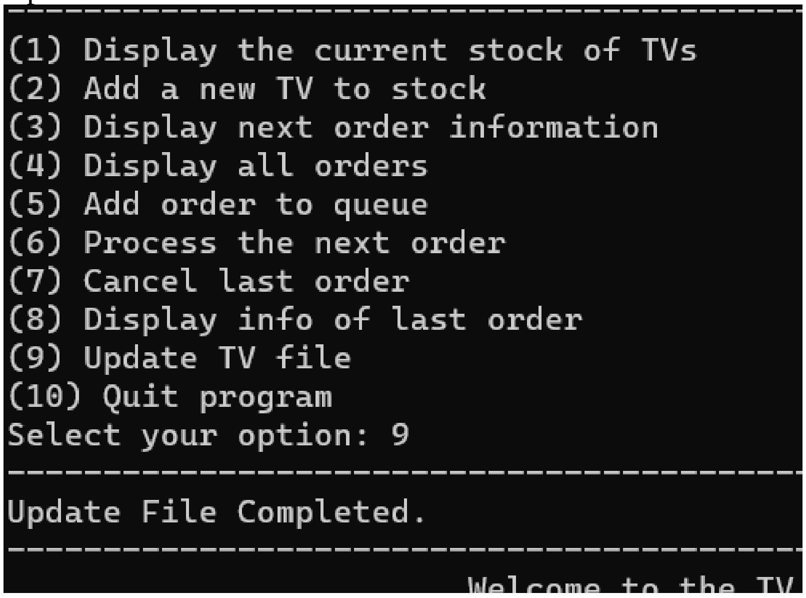
```
(10) Quit program
Select your option: 1
------------------------------
ID      Product Name
------------------------------
105     Apple LCD 2020
109     Apple TV
432     Baolo
102     LG G2 FHD
103     Samsung S95B LCD
107     Samsung The Frame
104     Sony A95K LCD
108     Sony XR X90K
106     TCL 6 Series
101     TCL U8H
110     Zen X1000
------------------------------
```

Option 8:



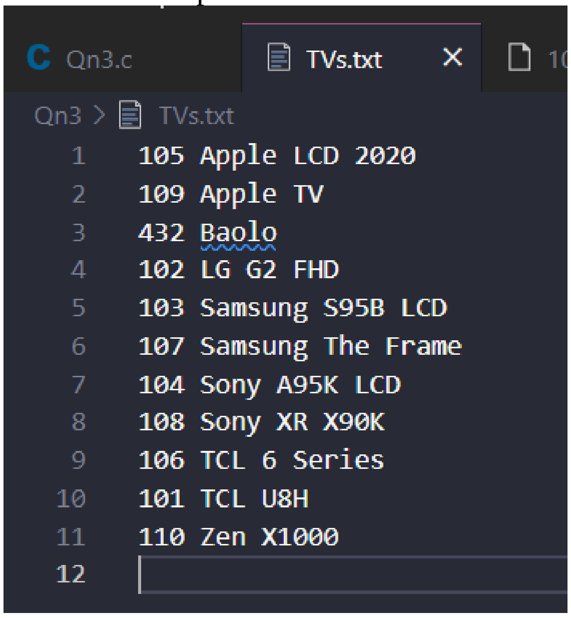
```
(10) Quit program
Select your option: 8
----------------------------------------
Information of last order is:
109     Apple TV
----------------------------------------
                         Welcome to the T
```

Option 9:



```
------------------------------------------
(1) Display the current stock of TVs
(2) Add a new TV to stock
(3) Display next order information
(4) Display all orders
(5) Add order to queue
(6) Process the next order
(7) Cancel last order
(8) Display info of last order
(9) Update TV file
(10) Quit program
Select your option: 9
------------------------------------------
Update File Completed.
------------------------------------------
                        Welcome to the TV
```

Text file also updated.



```
105 Apple LCD 2020
109 Apple TV
432 Baolo
102 LG G2 FHD
103 Samsung S95B LCD
107 Samsung The Frame
104 Sony A95K LCD
108 Sony XR X90K
106 TCL 6 Series
101 TCL U8H
110 Zen X1000

```