TODO LIST APP USING C

Problem Statement:

The problem addressed in this project is the need for a simple and efficient Todo List application. The application aims to provide users with a platform to manage their tasks by allowing them to create, mark as completed, edit, and view todos based on specific dates or as a complete list. The main challenge is to implement the required functionality while ensuring a user-friendly interface and efficient data management.

Algorithm:

The Todo List application utilizes a linked list data structure to store and manage the todos. The following algorithms are implemented:

a) `createNode(item)`: This algorithm creates a new node with the given todo item and returns a pointer to the newly created node.

b) `insertNode(head, item)`: This algorithm inserts a new node with the given todo item at the end of the linked list.

c) `markCompleted(head, index)`: This algorithm marks a task as completed by setting the `completed` flag of the corresponding node to 1.

d) `editTask(head, index)`: This algorithm allows the user to edit the task of a specific todo item by updating the corresponding node in the linked list.

e) `bubbleSort(head)`: This algorithm sorts the linked list in ascending order based on the dates of the todo items, using the bubble sort algorithm.

f) `compareDates(item1, item2)`: This algorithm compares two todo items based on their dates and returns a value indicating their relative order.

g) `displayTodoList(head, day, month, year)`: This algorithm displays the todos for a specific date, sorted by date.

h) `displayAllTodos(head)`: This algorithm displays all the todos in the linked list, sorted by date.

i) `freeList(head)`: This algorithm frees the memory occupied by the linked list nodes.

j) `storeTodoListToFile(head)`: This algorithm stores the todo list in a file named 'todo.txt'.

Input/Output:

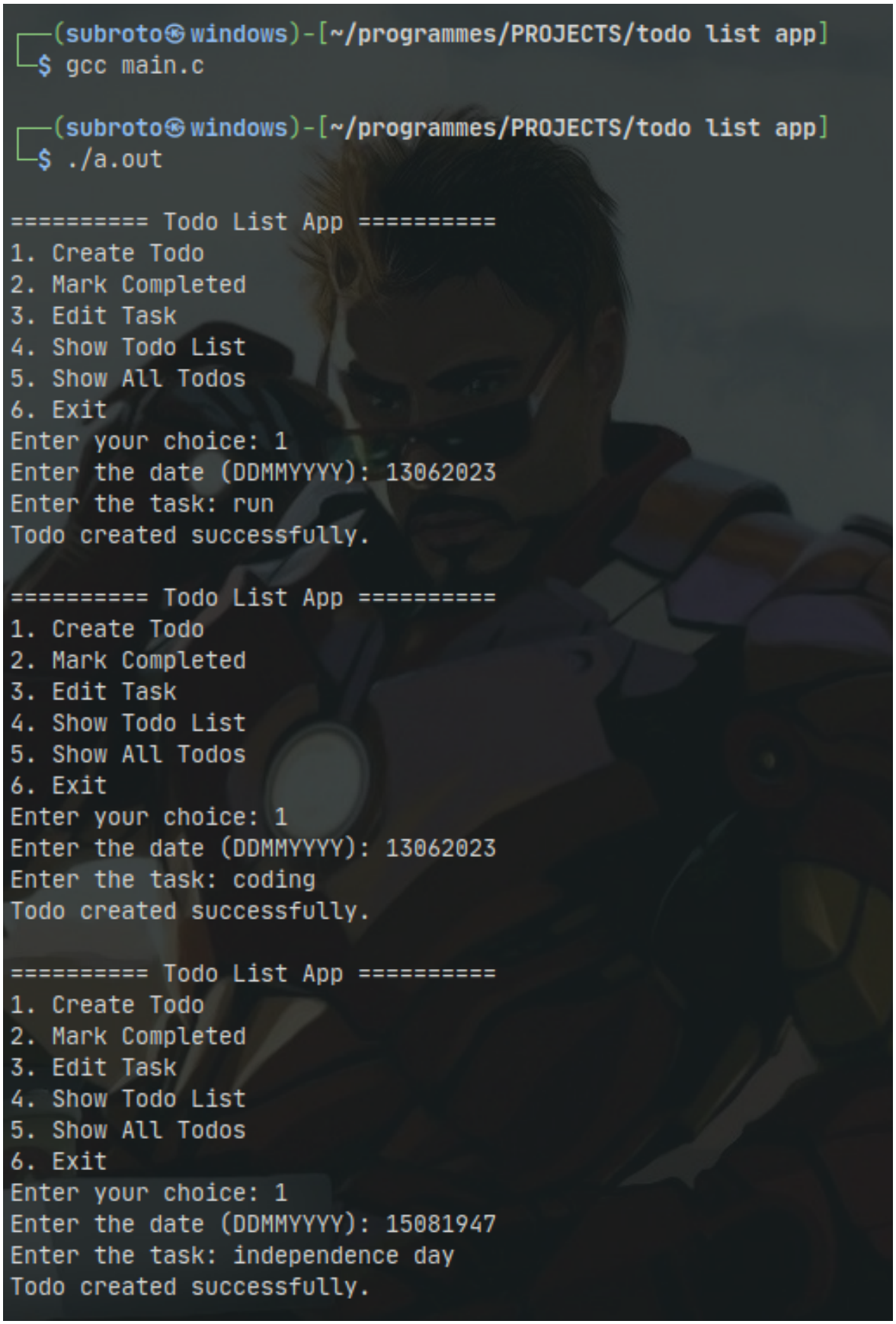
The input to the application is provided through the user interface, which presents a menu of options to the user. The user can choose from the following options:

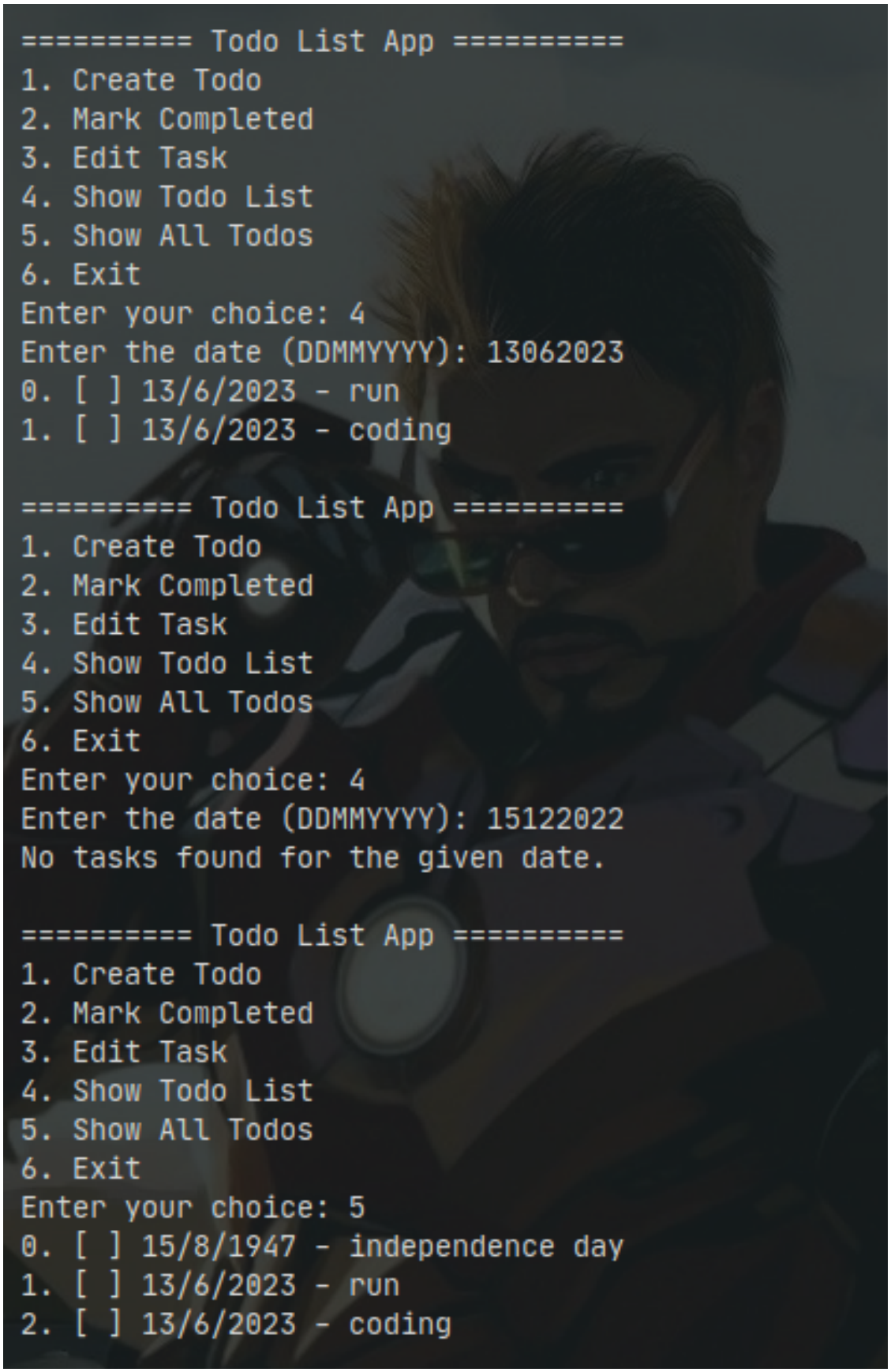
1. Create Todo: The user can enter the date and task details to create a new todo item.
2. Mark Completed: The user can select a todo item from the list and mark it as completed.
3. Edit Task: The user can select a todo item from the list and edit its task details.
4. Show Todo List: The user can enter a specific date and view the todos for that date.
5. Show All Todos: The user can view all the todos in the list, sorted by date.
6. Exit: The user can choose to exit the application.

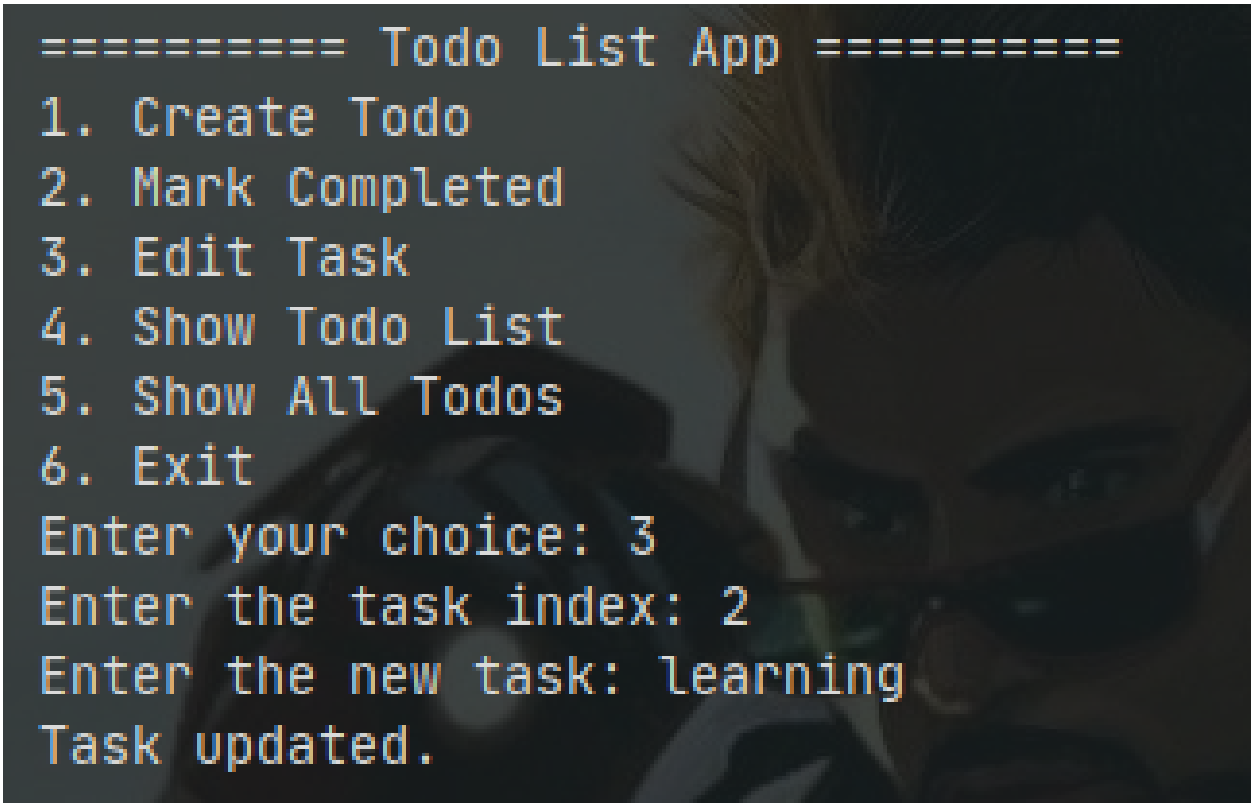
The output of the application is displayed on the console. It includes notifications and information about the status of operations, such as the successful creation of a todo,

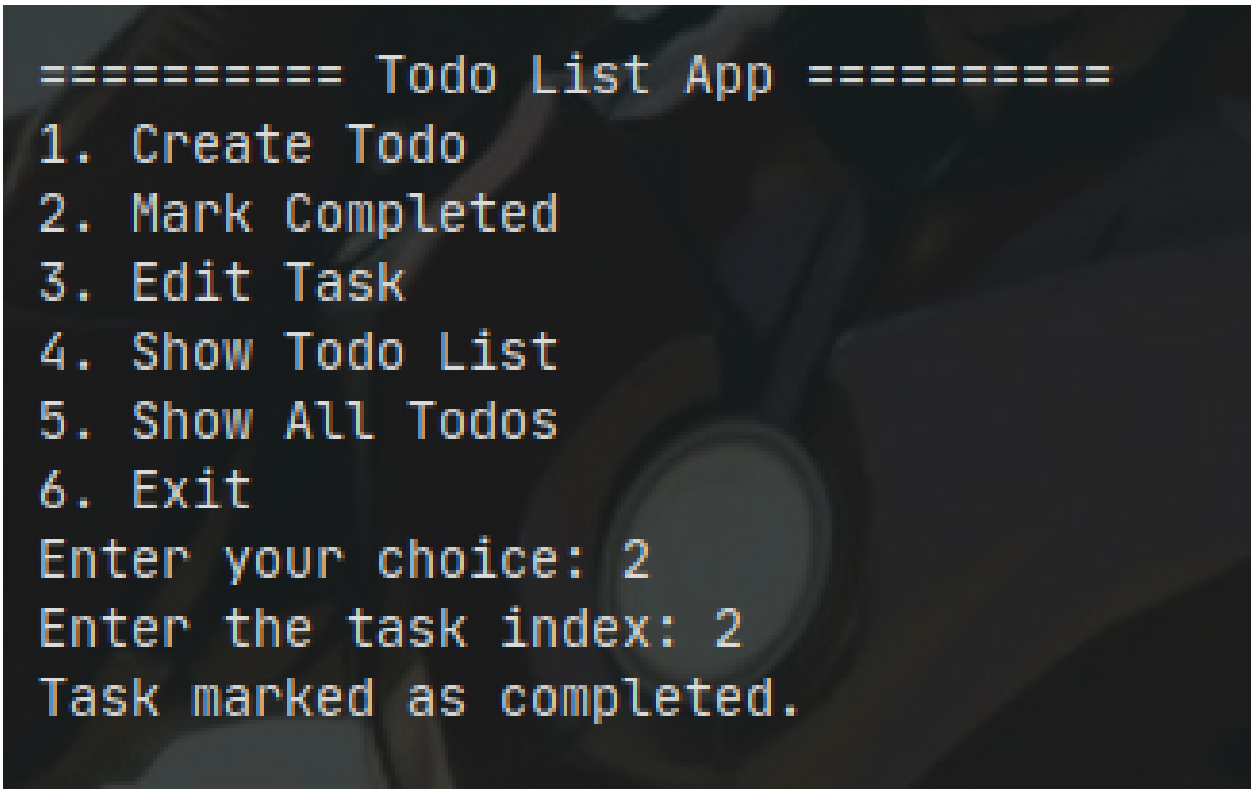
marking a task as completed, updating a task, displaying todos for a specific date, storing the todo list in a file, and exiting the program.

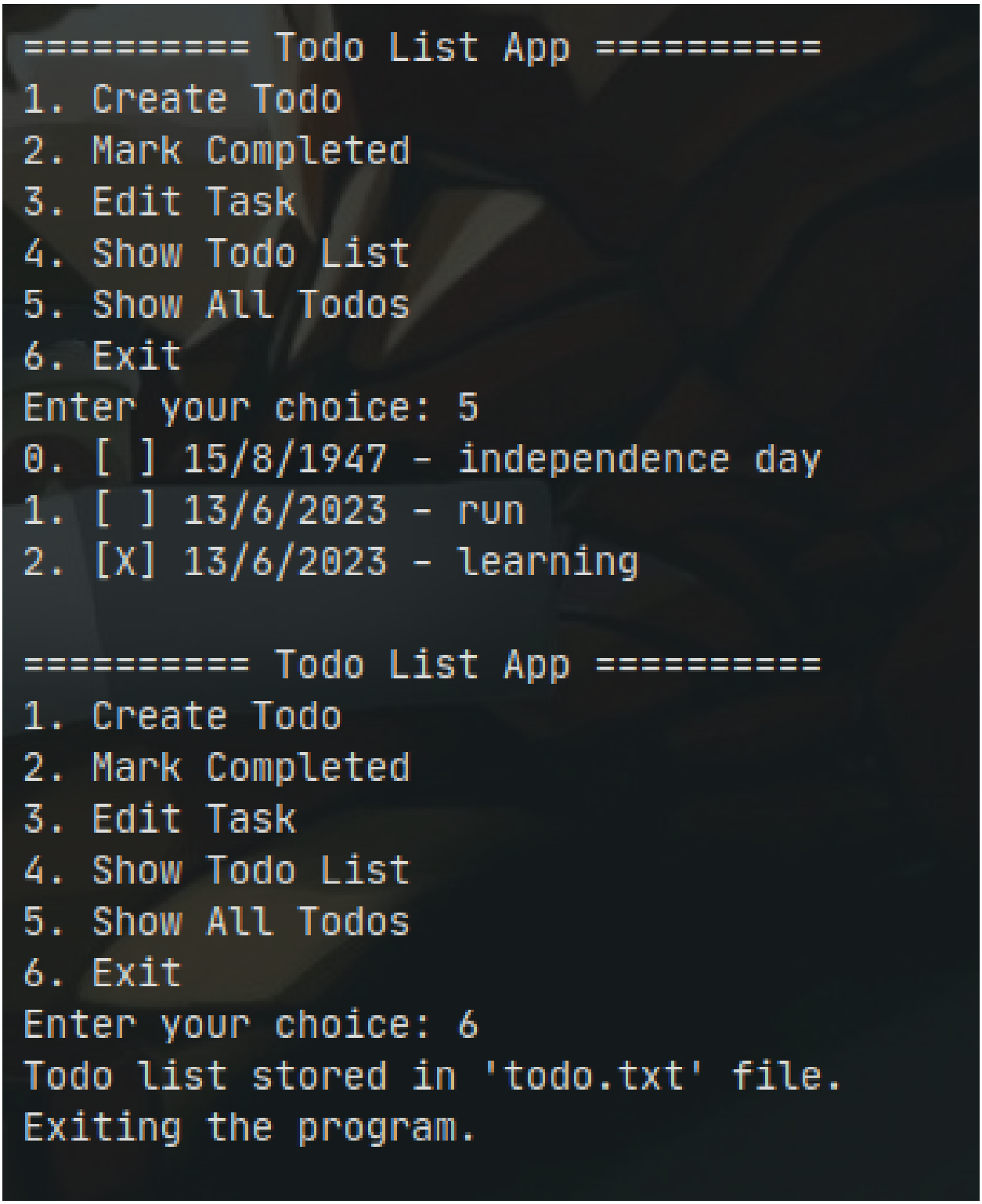
Input/Output Screenshots

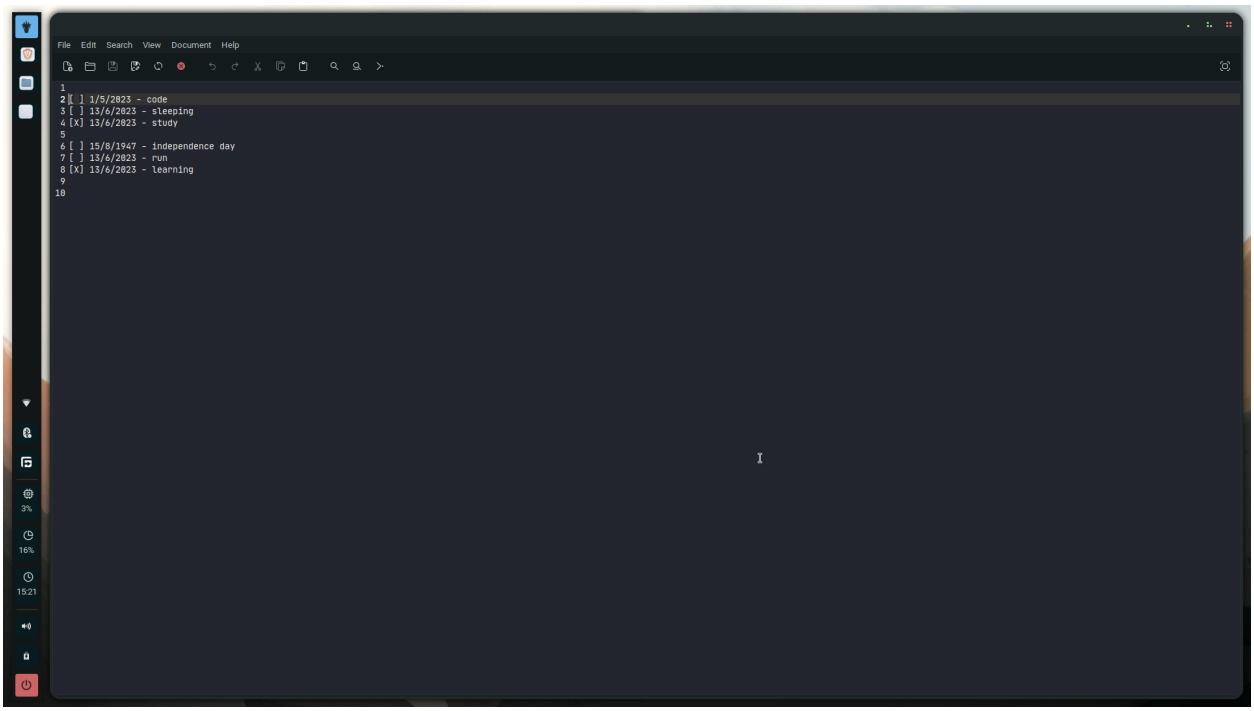

(subroto㉿windows)-[~/programmes/PROJECTS/todo list app]
$ gcc main.c
(subroto㉿windows)-[~/programmes/PROJECTS/todo list app]
$ ./a.out
========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 1
Enter the date (DDMMYYYY): 13062023
Enter the task: run
Todo created successfully.
========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 1
Enter the date (DDMMYYYY): 13062023
Enter the task: coding
Todo created successfully.
========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 1
Enter the date (DDMMYYYY): 15081947
Enter the task: independence day
Todo created successfully.


========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 4
Enter the date (DDMMYYYY): 13062023
0. [ ] 13/6/2023 - run
1. [ ] 13/6/2023 - coding
========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 4
Enter the date (DDMMYYYY): 15122022
No tasks found for the given date.
========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 5
0. [ ] 15/8/1947 - independence day
1. [ ] 13/6/2023 - run
2. [ ] 13/6/2023 - coding


========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 3
Enter the task index: 2
Enter the new task: learning
Task updated.


========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 2
Enter the task index: 2
Task marked as completed.


========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 5
0. [ ] 15/8/1947 - independence day
1. [ ] 13/6/2023 - run
2. [X] 13/6/2023 - learning
========== Todo List App ==========
1. Create Todo
2. Mark Completed
3. Edit Task
4. Show Todo List
5. Show All Todos
6. Exit
Enter your choice: 6
Todo list stored in 'todo.txt' file.
Exiting the program.



Discussions:

a) Limitations:

- The current implementation of the Todo List application has a command-line interface, which may limit its accessibility for users who prefer graphical interfaces.
- The application does not support the deletion of specific todo items from the list. Once a task is completed, it remains in the list, marked as completed.
- The sorting algorithm used in the application is the bubble sort algorithm, which may not be the most efficient for large lists. Implementing a more efficient sorting algorithm could improve the performance of the application.
- The application does not provide any form of authentication or user management, which may limit its usage to a single user on a local machine.

b) Future Scope of Work:

- Enhancing the user interface by developing a graphical interface or a web-based interface to make the application more user-friendly and accessible.

- Implementing advanced sorting algorithms, such as merge sort or quicksort, to improve the performance of sorting large lists of todos.
- Adding the ability to delete specific todo items from the list, giving users more flexibility in managing their tasks.
- Implementing user authentication and user management features, allowing multiple users to have personalized todo lists and ensuring data security.
- Providing notifications or reminders for upcoming tasks or deadlines, helping users stay organized and meet their goals.
- Adding the ability to categorize tasks or prioritize them, allowing users to manage different types of tasks effectively.
- Integrating cloud storage or synchronization functionality, enabling users to access their todo lists across multiple devices.
- Implementing data backup and recovery mechanisms to prevent data loss and ensure the safety of users' todo lists.

In conclusion, the Todo List application provides a simple yet effective solution for managing tasks. While it has certain limitations, such as the command-line interface and the absence of certain features, there is a wide scope for further improvements and enhancements. With the integration of user-friendly interfaces, advanced sorting algorithms, additional features, and data security measures, the Todo List application can become a comprehensive and versatile tool for individuals and teams to manage their tasks efficiently.