SECJ2013: Data Structure and Algorithm

Faculty of Computing

Task Management System Project

Group members

Name	Matrics Number
Muhammad Luqman Hakim Bin Mohd Rizaudin	A22EC0086
Muhammad Anas Bin Mohd Pikri	A21SC0464
Kugen a/l Kalidas	A22EC0178

Prepared by: Tempest

Submitted date : 16/12/2023

Table of Contents

1. **Objective**

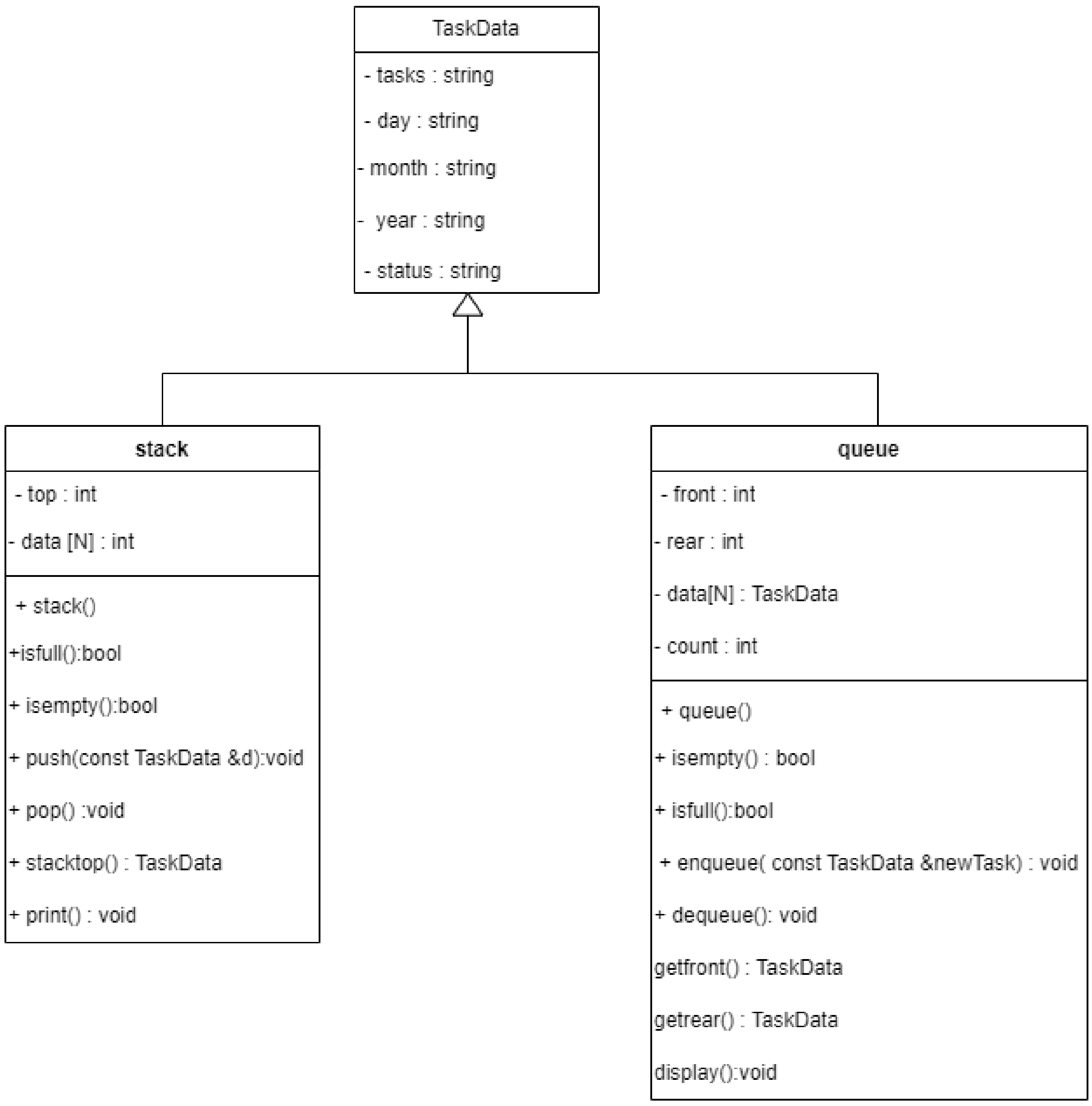
The stack technique is used in this task management system to facilitate user navigation by preserving a history stack that allows users to easily backtrack through their interactions. A stack is also used to implement a reverse feature, which allows users to return to previous states. The queue technique, on the other hand, is used to manage tasks in a first-in, first-out manner, ensuring task execution fairness. Furthermore, an update queue is implemented, which uses a queue to organise and present updates to users in a systematic manner. These techniques, when combined, improve data accessibility and user experience in the system.

2. Synopsis

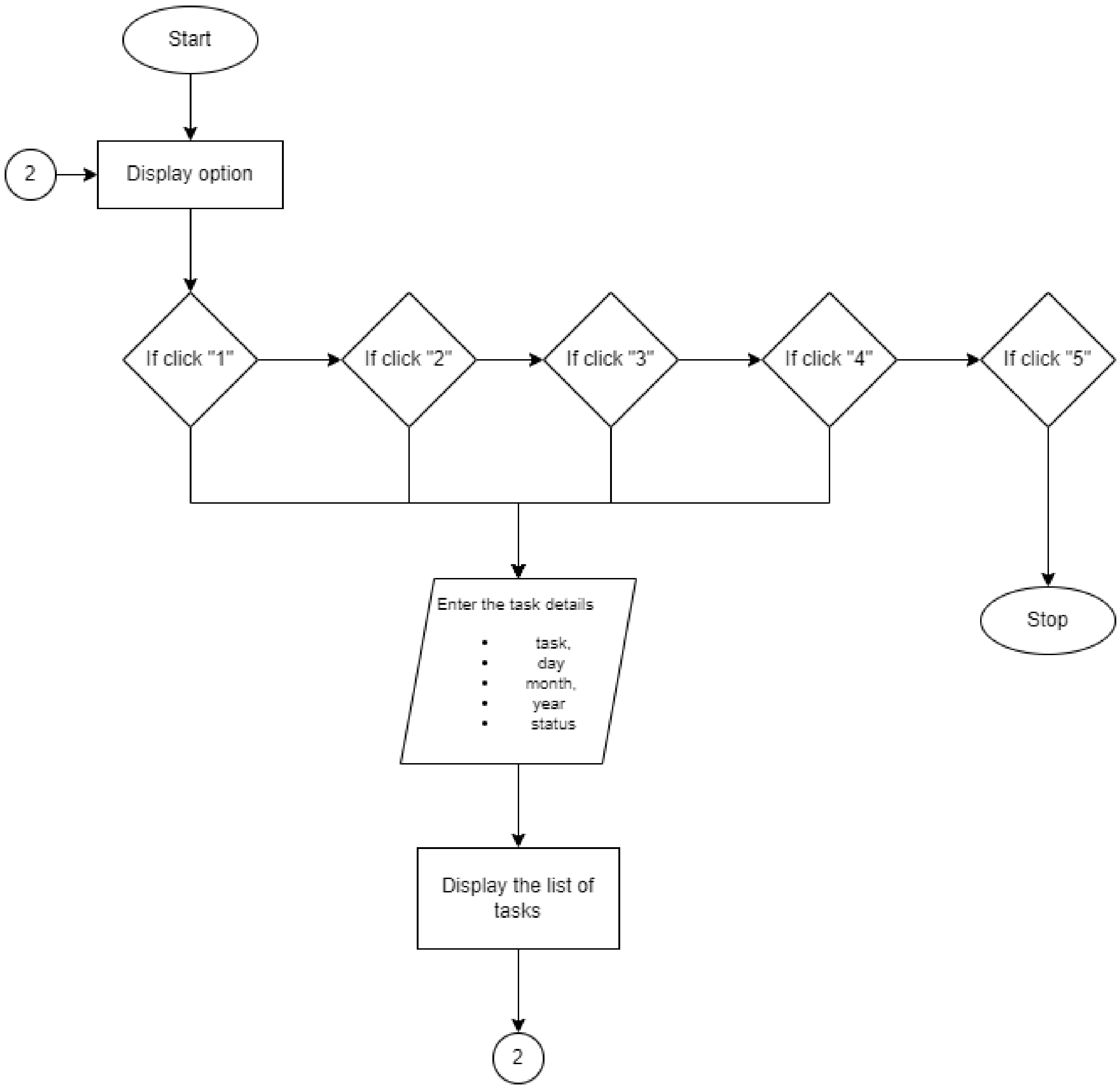
Users encounter a comprehensive display of tasks, which includes dates and statuses, in the task management system's main menu, as well as an option menu with distinct options such as stack and queue. When users select the stack option, they are redirected to a sub-menu designed for last-in-first-out (LIFO) task data manipulation. Users can add tasks (push them onto the stack), remove tasks (pop them off the stack), and inspect tasks in a stacked manner, providing a dynamic and simple approach to task management. In the same way, selecting the queue option directs users to a sub-menu designed for first-in-first-out (FIFO) task manipulation. This menu contains options for adding tasks (enqueuing), removing tasks (dequeuing), and examining tasks in a queued manner, introducing a systematic and ordered approach to task management.

3. Design

3.1 Class Diagram



3.2 Flowchart of Task Management System



4. Description of Data Structure Operation: Stack and Queue

1. User will be directed to the main menu of the task management system. In the main menu user can see all the task with its date and status. User will also can see the option menu that consists of stack and queue data operation methods.

2. If the user click on the option 1 which add task in stack method, the system will show the output where the task will be added on the top of the list

3. The user will go to the main menu and will be showed again the first event. There the user can choose again which option that the user wants.

4. If the user click on the option 2 which delete task in stack method, the system will show the output where the task will be deleted on the top of the list.

5. The user will go to the main menu and will be showed again the first event. There the user can choose again which option that the user wants.

6. If the user click on the option 3 which add task in queue method, the system will show the output where the task will be added on the last of the list.

7. The user will go to the main menu and will be showed again the first event. There the user can choose again which option that the user wants.

8. If the user click on the option 4 which delete task in queue method, the system will show the output where the task will be deleted at the first of the list.

9. The user will go to the main menu and will be showed again the first event. There the user can choose again which option that the user wants.

10. If the user click on the option 5 which exit, the system will end the session and user will exiting the system.

5. Source Code Demonstrating The Data Structure Concept Employed

5.1 Stack Concept Code

```
class stack
{
private:
    int top;
    TaskData data[N]; // Array of structures to store task details

public:
    stack()
    {
        top = -1;
    };

    bool isfull()
    {
        return top == N - 1;
    }

    bool isempty()
    {
        return top == -1;
    }

    void push(const TaskData &d)
    {
        if (isfull())
            cout << "Stack is full" << endl;

        else
        {
            top++;
            data[top] = d;
        }
    }

    void pop()
    {
        if (isempty())
        {
            cout << "Stack is empty" << endl;
        }

        else
            top--;
    }

    TaskData stacktop()
    {
        return data[top];
    }

    void print()
    {
        if (isempty())
        {
            cout << "sorry, stack is empty" << endl;
        }
        else
        {
            for (int a = top; a >= 0; a--)
            {
                cout << left << setw(40) << data[a].tasks;
                cout << left << setw(2) << "|";
                cout << left << setw(20) << data[a].day + "-" +
data[a].month + "-" + data[a].year;
                cout << left << setw(2) << "|";
                cout << left << setw(10) << data[a].status << endl;

                cout << endl;
            }
        }
    }
};
```

```
case 1:
        {
            cout << "Enter new task details:" << endl;
            TaskData newTask;
            cout << "Task: ";
            cin >> newTask.tasks;
            cout << "Day: ";
            cin >> newTask.day;
```

```cpp
            cout << "Month: ";
            cin >> newTask.month;
            cout << "Year: ";
            cin >> newTask.year;
            cout << "Status: ";
            cin >> newTask.status;

            s.push(newTask);
            // Display tasks using stack method
            cout << "\nTasks using Stack method:" << endl;
            cout << left << setw(40) << "TASK";
            cout << left << setw(2) << "|";
            cout << left << setw(20) << "DATE";
            cout << left << setw(2) << "|";
            cout << left << setw(10) << "STATUS" << endl
                 << endl;
            // Display the entire stack
            s.print();
            cout << endl;
        }
        break;

        case 2:
        {
            s.pop();
            // Display tasks using stack method
            cout << "\nTasks using Stack method:" << endl;
            cout << left << setw(40) << "TASK";
            cout << left << setw(2) << "|";
            cout << left << setw(20) << "DATE";
            cout << left << setw(2) << "|";
            cout << left << setw(10) << "STATUS" << endl
                 << endl;
            // Display the entire stack
            s.print();
            cout << endl;
        }
        break;
```

5.2 Queue Concept Code

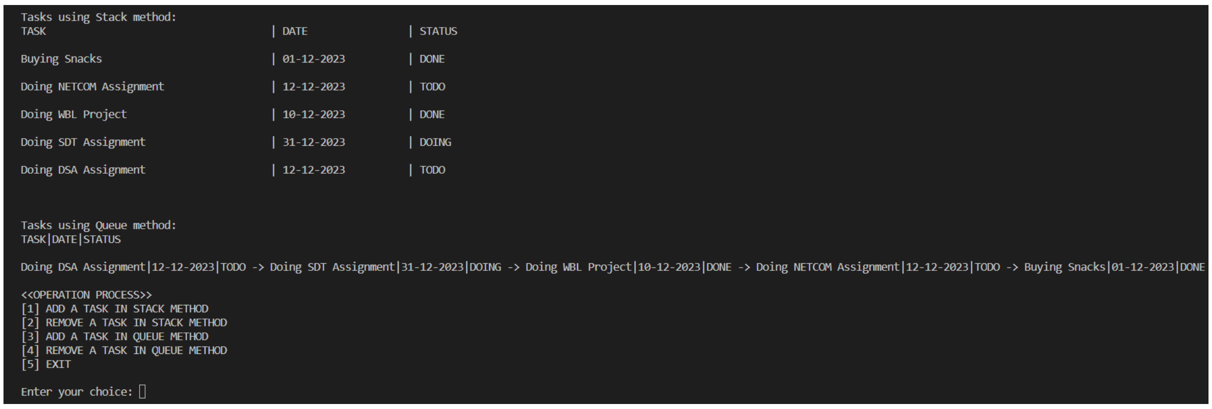
```
class queue
{
private:
    int front;
    int rear;
    TaskData data[N];
    int count;

public:
    queue()
    {
        front = 0;
        rear = N - 1;
        count = 0;
    }

    bool isempty()
    {
        return count == 0;
    }

    bool isfull()
    {
        return count == N;
    }

    void enqueue(const TaskData &newTask)
    {
        if (isfull())
        {
            cout << "Sorry, the queue is full" << endl;
        }
        else
        {
            rear = (rear + 1) % N; //*
            data[rear] = newTask;
            count++;
        }
    }

    void dequeue()
    {
        if (isempty())
        {
            cout << "Sorry, the queue is empty" << endl;
        }
        else
        {
            front = (front + 1) % N;
            count--;
        }
    }

    TaskData getfront()
    {
        return data[front];
    }

    TaskData getrear()
    {
        return data[rear];
    }

    void display()
    {
        if (isempty())
        {
            cout << "queue is empty" << endl;
        }
        else if (front <= rear)
        {
            for (int a = front; a < rear; a++)
            {
                cout << data[a].tasks << "|" << data[a].day + "-" +
data[a].month + "-" + data[a].year << "|" << data[a].status << " -> ";
            }
            // Display the last element without the arrow
            cout << data[rear].tasks << "|" << data[rear].day + "-" +
data[rear].month + "-" + data[rear].year << "|" << data[rear].status;
        }
        else
        {
            for (int a = front; a < N - 1; a++)
            {
                cout << data[a].tasks << "|" << data[a].day + "-" +
data[a].month + "-" + data[a].year << "|" << data[a].status << "->";
            }
            // Display the last element without the arrow
            cout << data[rear].tasks << "|" << data[rear].day + "-" +
data[rear].month + "-" + data[rear].year << "|" << data[rear].status;
        }
        cout << endl;
    }
};
```

```
case 3:
        {
            cout << "Enter new task details:" << endl;
            TaskData newTask;
            cout << "Task: ";
            cin >> newTask.tasks;
            cout << "Day: ";
            cin >> newTask.day;
            cout << "Month: ";
            cin >> newTask.month;
            cout << "Year: ";
            cin >> newTask.year;
            cout << "Status: ";
            cin >> newTask.status;

            q.enqueue(newTask);
            // Display tasks using queue method
            cout << "\nTasks using Queue method:" << endl;
            cout << "TASK|DATE|STATUS" << endl
                 << endl;
            // Display the entire queue
            q.display();
            cout << endl;
        }
        break;

        case 4:
        {
            q.dequeue();
            // Display tasks using queue method
            cout << "\nTasks using Queue method:" << endl;
            cout << "TASK|DATE|STATUS" << endl
```

```
                << endl;
            // Display the entire queue
            q.display();
            cout << endl;
        }
        break;
```

6. User Manual/Guide

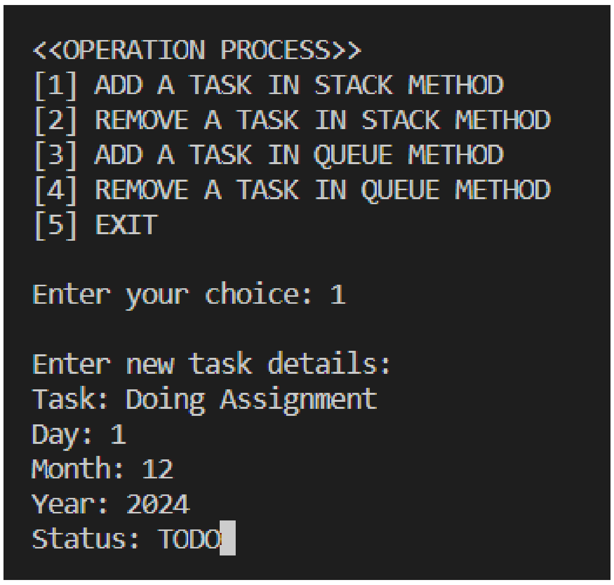
1. When the user enter the Task management system, the system will show the main page with the task in stack and queue manner. The system will also give option to the user.



```
Tasks using Stack method:
TASK                              | DATE              | STATUS

Buying Snacks                     | 01-12-2023        | DONE

Doing NETCOM Assignment           | 12-12-2023        | TODO

Doing WBL Project                 | 10-12-2023        | DONE

Doing SDT Assignment              | 31-12-2023        | DOING

Doing DSA Assignment              | 12-12-2023        | TODO



Tasks using Queue method:
TASK|DATE|STATUS

Doing DSA Assignment|12-12-2023|TODO -> Doing SDT Assignment|31-12-2023|DOING -> Doing WBL Project|10-12-2023|DONE -> Doing NETCOM Assignment|12-12-2023|TODO -> Buying Snacks|01-12-2023|DONE

<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice:
```

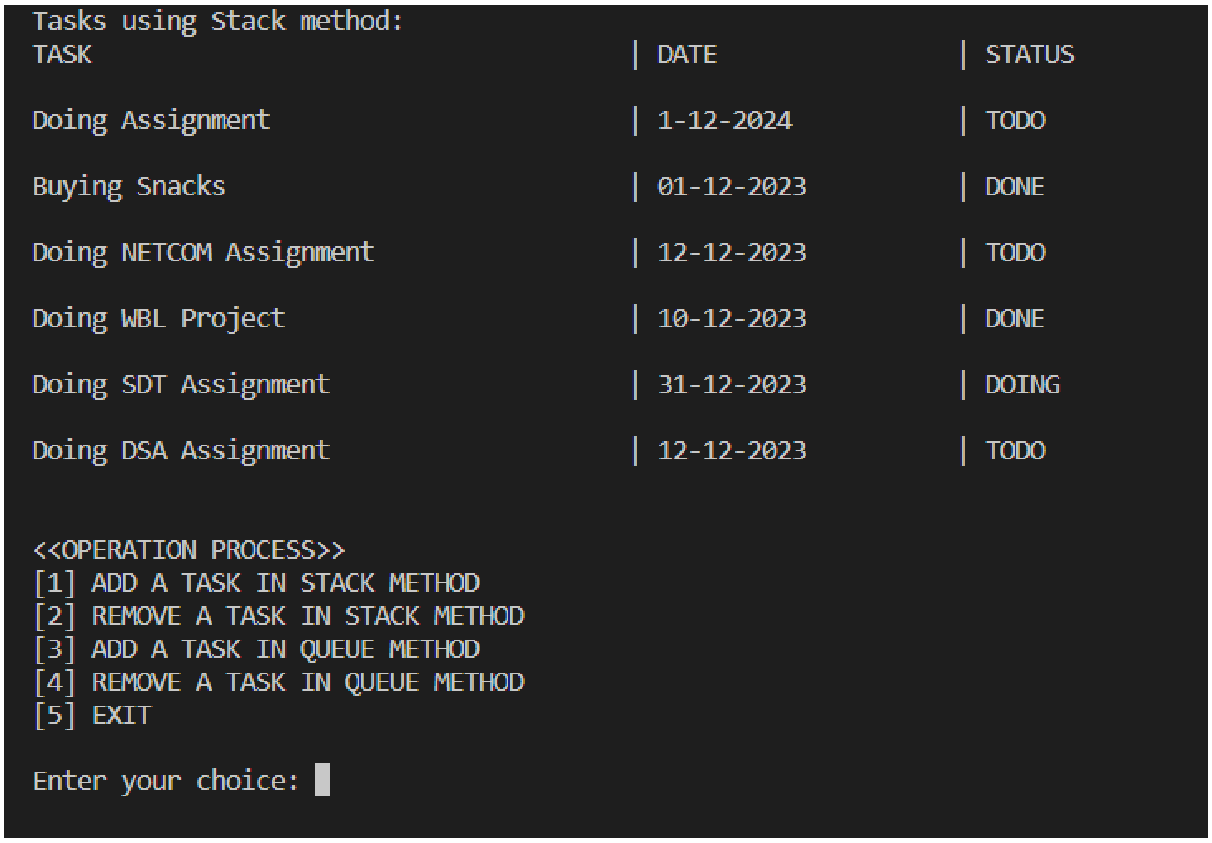
2. Then, in the operation process option, the user can choose 5 options which are add and delete task in stack method, add and delete task in queue method and exit the program.
3. If the user enter the option 1 which is the add a task in stack method, the user can enter the task detail which are task, date, and status. After insert the data, the system will shows the output which the new task will be added at the top of the list. Then the user will shown again the option to choose.

Input:



```
<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice: 1

Enter new task details:
Task: Doing Assignment
Day: 1
Month: 12
Year: 2024
Status: TODO
```

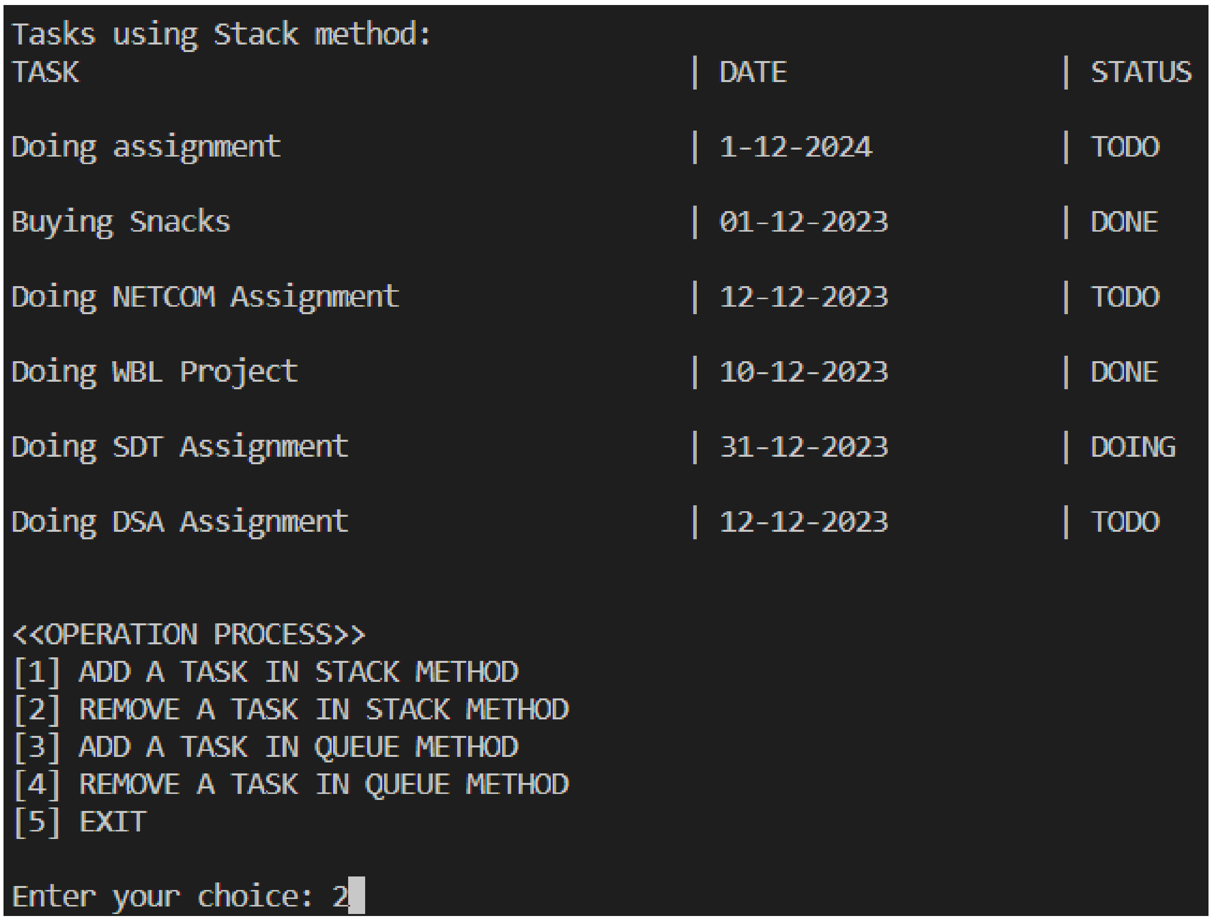
Output:



```
Tasks using Stack method:
TASK                                      | DATE                    | STATUS

Doing Assignment                          | 1-12-2024               | TODO

Buying Snacks                             | 01-12-2023              | DONE

Doing NETCOM Assignment                   | 12-12-2023              | TODO

Doing WBL Project                         | 10-12-2023              | DONE

Doing SDT Assignment                      | 31-12-2023              | DOING

Doing DSA Assignment                      | 12-12-2023              | TODO


<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice:
```

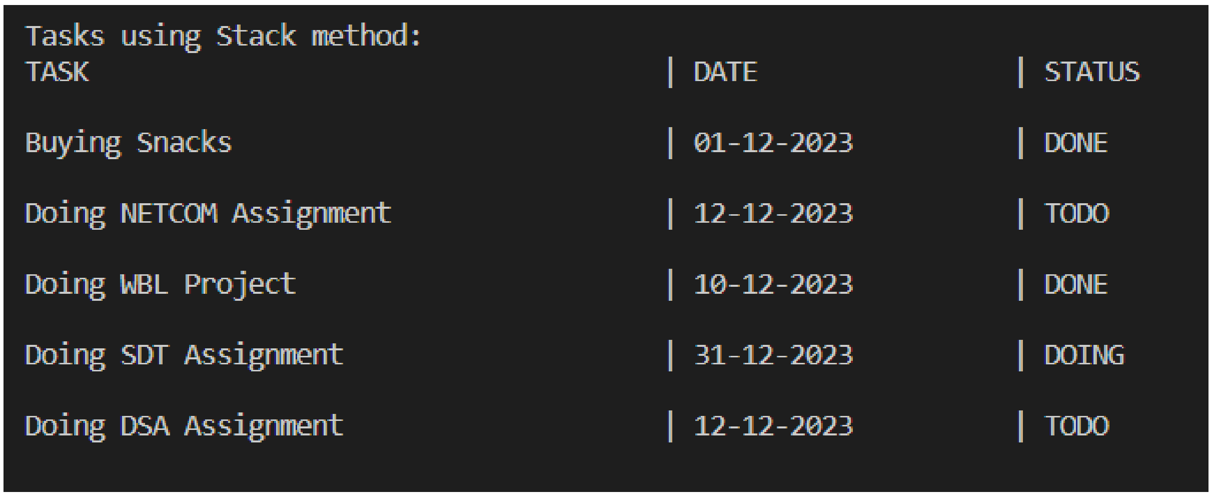
4. If the user enter the option 2 which is the delete a task in stack method, The system will delete the topmost task from the list. After delete the data, the system will shows the output which the topmost task will be deleted from the list. Then the user will shown again the option to choose.

Before:



```
Tasks using Stack method:
TASK                                     | DATE                   | STATUS

Doing assignment                         | 1-12-2024              | TODO

Buying Snacks                            | 01-12-2023             | DONE

Doing NETCOM Assignment                  | 12-12-2023             | TODO

Doing WBL Project                        | 10-12-2023             | DONE

Doing SDT Assignment                     | 31-12-2023             | DOING

Doing DSA Assignment                     | 12-12-2023             | TODO


<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice: 2
```

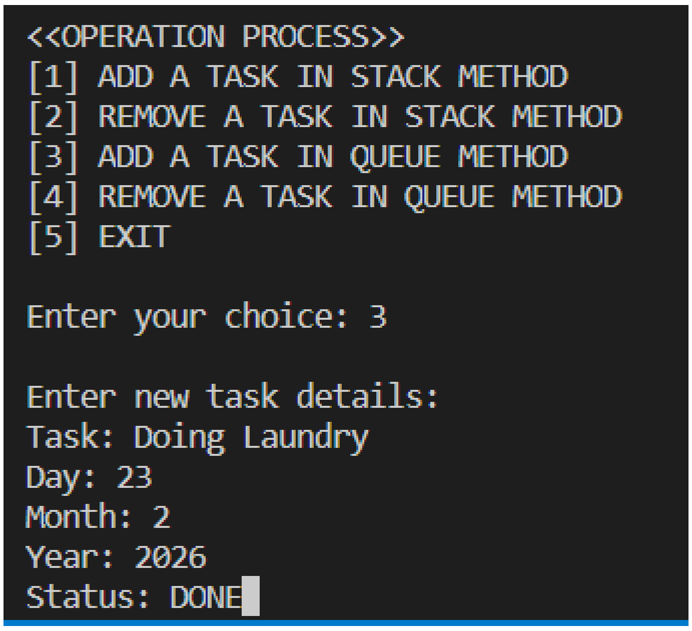
After:



```
Tasks using Stack method:
TASK                                     | DATE                   | STATUS

Buying Snacks                            | 01-12-2023             | DONE

Doing NETCOM Assignment                  | 12-12-2023             | TODO

Doing WBL Project                        | 10-12-2023             | DONE

Doing SDT Assignment                     | 31-12-2023             | DOING

Doing DSA Assignment                     | 12-12-2023             | TODO
```

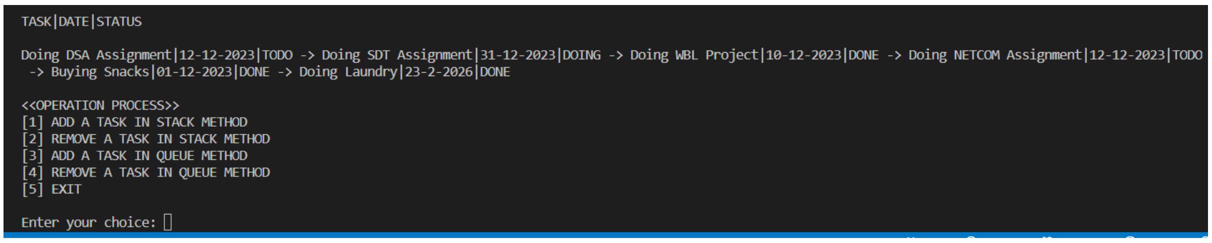
5. If the user enter the option 3 which is the add a task in queue method, the user can enter the task detail which are task, date, and status. After insert the data, the system will shows the output which the new task will be added at the end of the list. Then the user will shown again the option to choose.

Input:



```
<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice: 3

Enter new task details:
Task: Doing Laundry
Day: 23
Month: 2
Year: 2026
Status: DONE
```

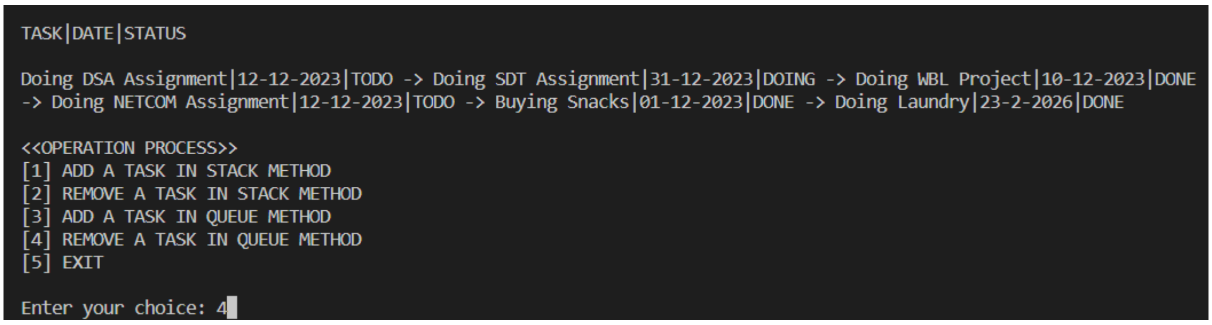
Output:



```
TASK|DATE|STATUS

Doing DSA Assignment|12-12-2023|TODO -> Doing SDT Assignment|31-12-2023|DOING -> Doing WBL Project|10-12-2023|DONE -> Doing NETCOM Assignment|12-12-2023|TODO
 -> Buying Snacks|01-12-2023|DONE -> Doing Laundry|23-2-2026|DONE

<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice: 
```

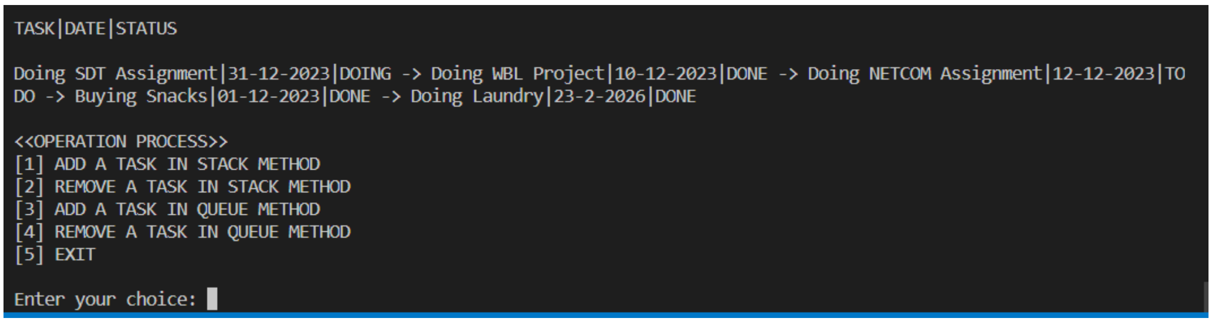
6. If the user enter the option 4 which is the delete a task in queue method, The system will delete the first task from the list. After delete the data, the system will shows the output which the first task will be deleted from the list. Then the user will shown again the option to choose.

Before:



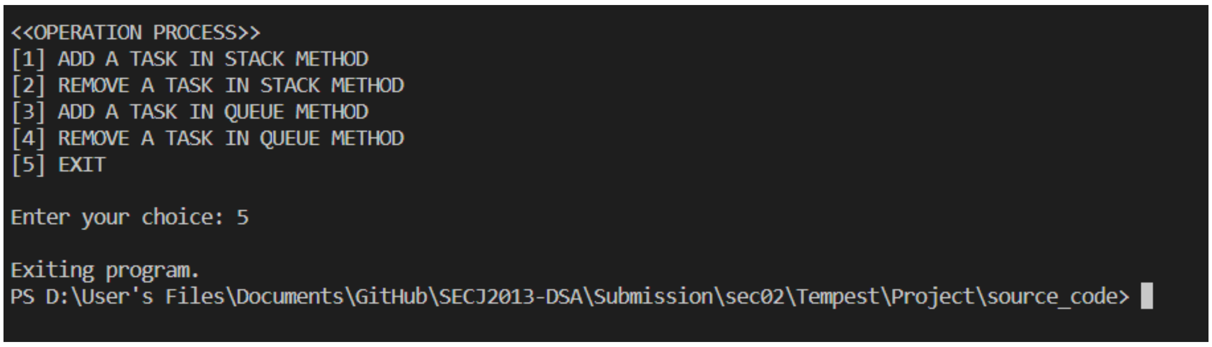
```
TASK|DATE|STATUS

Doing DSA Assignment|12-12-2023|TODO -> Doing SDT Assignment|31-12-2023|DOING -> Doing WBL Project|10-12-2023|DONE
-> Doing NETCOM Assignment|12-12-2023|TODO -> Buying Snacks|01-12-2023|DONE -> Doing Laundry|23-2-2026|DONE

<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice: 4
```

After:



```
TASK|DATE|STATUS

Doing SDT Assignment|31-12-2023|DOING -> Doing WBL Project|10-12-2023|DONE -> Doing NETCOM Assignment|12-12-2023|TO
DO -> Buying Snacks|01-12-2023|DONE -> Doing Laundry|23-2-2026|DONE

<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice:
```

7. If the user enter the option 5 which exit, the system will stop the session and the user will be exiting the system.



```
<<OPERATION PROCESS>>
[1] ADD A TASK IN STACK METHOD
[2] REMOVE A TASK IN STACK METHOD
[3] ADD A TASK IN QUEUE METHOD
[4] REMOVE A TASK IN QUEUE METHOD
[5] EXIT

Enter your choice: 5

Exiting program.
PS D:\User's Files\Documents\GitHub\SECJ2013-DSA\Submission\sec02\Tempest\Project\source_code>
```