**Week 2**

**Write Programs to implement Stack, Queue Operations using Linked List**

1. **C Programs to implement Stack Operations using Linked List**

#include <stdio.h>

#include <stdlib.h>

*// Define node structure*

struct **Node**

{

int data;

struct Node \*next;

};

*// Top pointer to keep track of the stack top*

struct Node \*top = NULL;

*// Push operation*

void **push**(int value)

{

struct Node \*newNode = (struct Node\*)malloc(sizeof(struct Node));

if (!newNode)

{

printf("Heap overflow! Unable to Push %d\n", value);

return;

}

newNode->data = value;

newNode->next = top;

top = newNode;

printf("%d Pushed into Stack\n", value);

}

*// Pop operation*

void **pop**()

{

if (top == NULL)

{

printf("Stack Underflow! Cannot Pop\n");

return;

}

struct Node \*temp = top;

printf("%d Popped from Stack\n", top->data);

top = top->next;

free(temp);

}

*// Peek operation*

void **peek**()

{

if (top == NULL)

{

printf("Stack is Empty\n");

}

else

{

printf("Top element: %d\n", top->data);

}

}

*// Display the stack*

void **display**()

{

struct Node \*temp = top;

printf("Top -> ");

while (temp != NULL)

{

printf("[%d] -> ", temp->data);

temp = temp->next;

}

printf("NULL\n");

}

*// Main menu-driven function*

int **main**()

{

int choice, value;

while (1)

{

printf("\n--- Stack Menu ---\n");

printf("1. Push\n2. Pop\n3. Peek\n4. Display\n5. Exit\n");

printf("Enter your Choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

printf("Enter value to Push: ");

scanf("%d", &value);

push(value);

break;

case 2:

pop();

break;

case 3:

peek();

break;

case 4:

display();

break;

case 5:

printf("Exiting Program.\n");

exit(0);

default:

printf("Invalid Choice! Try again.\n");

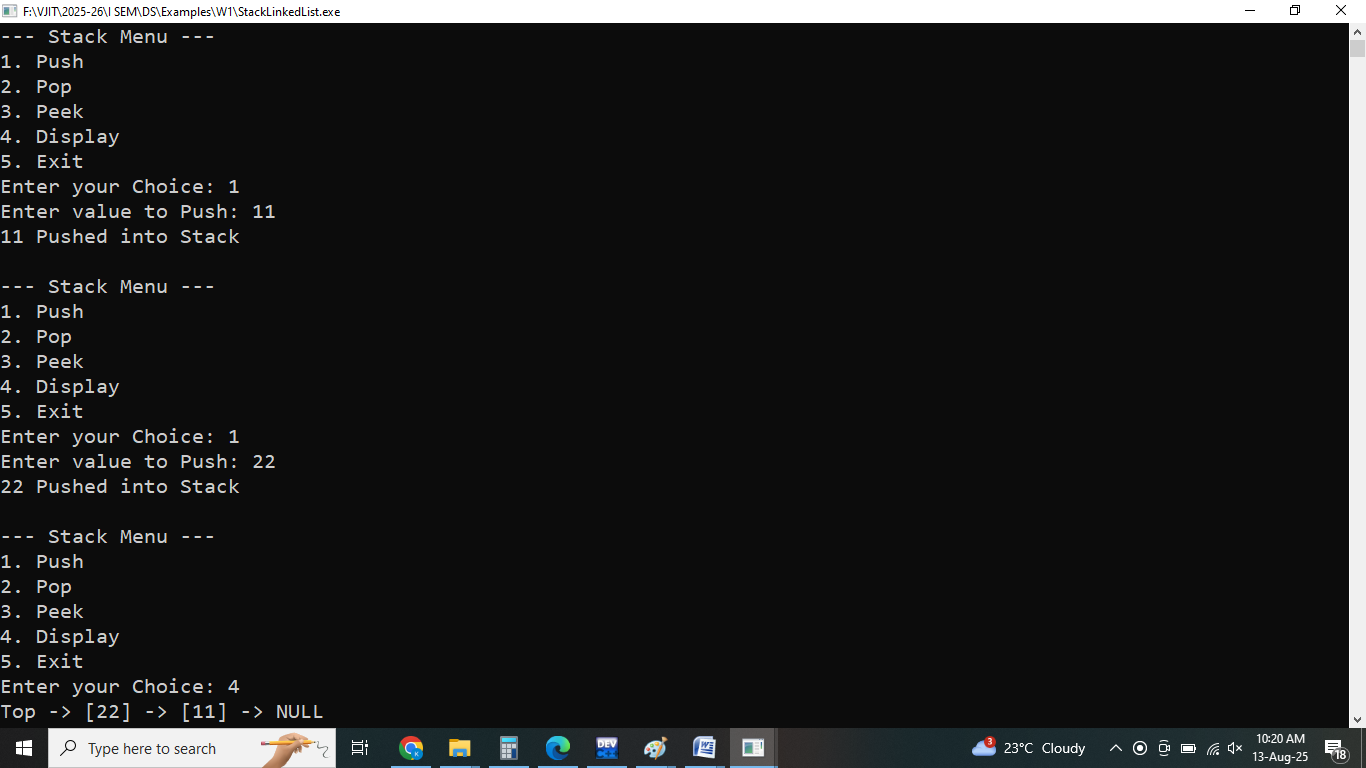
}

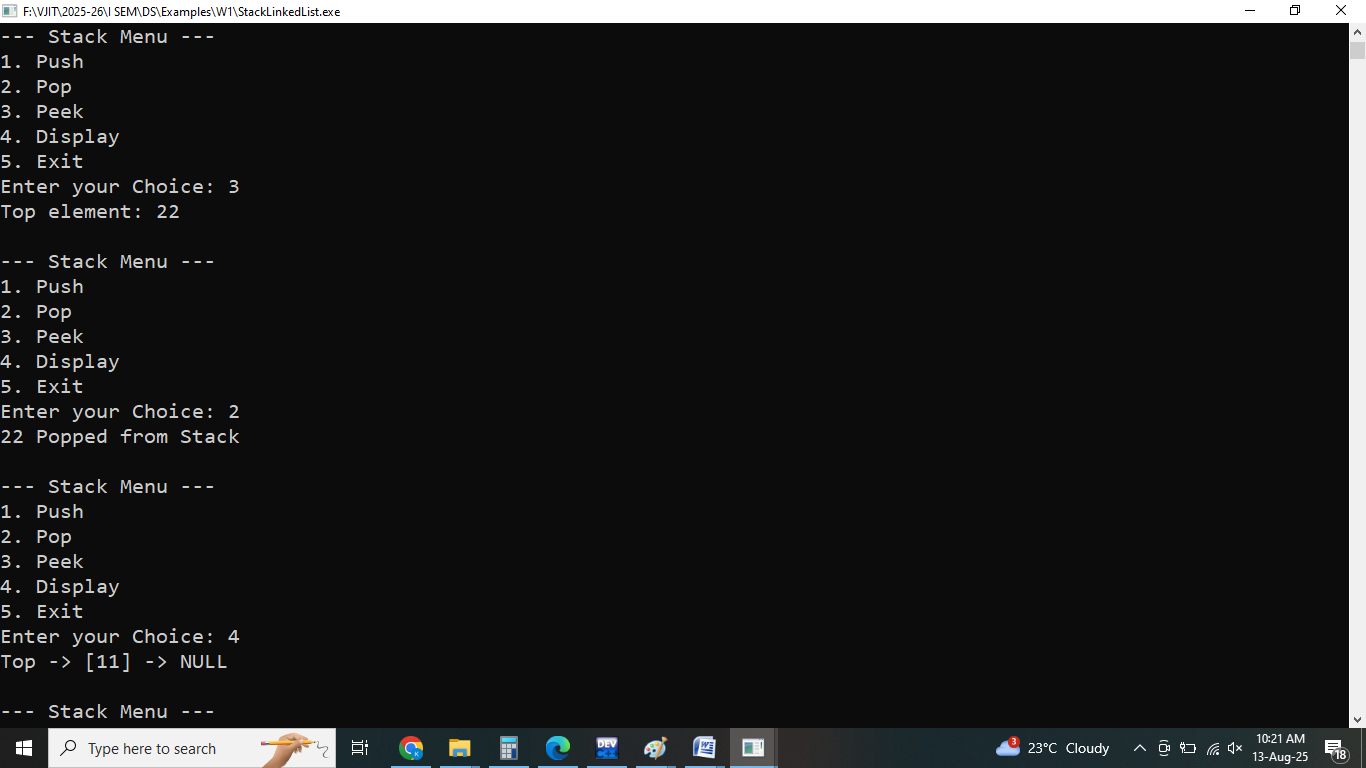
}

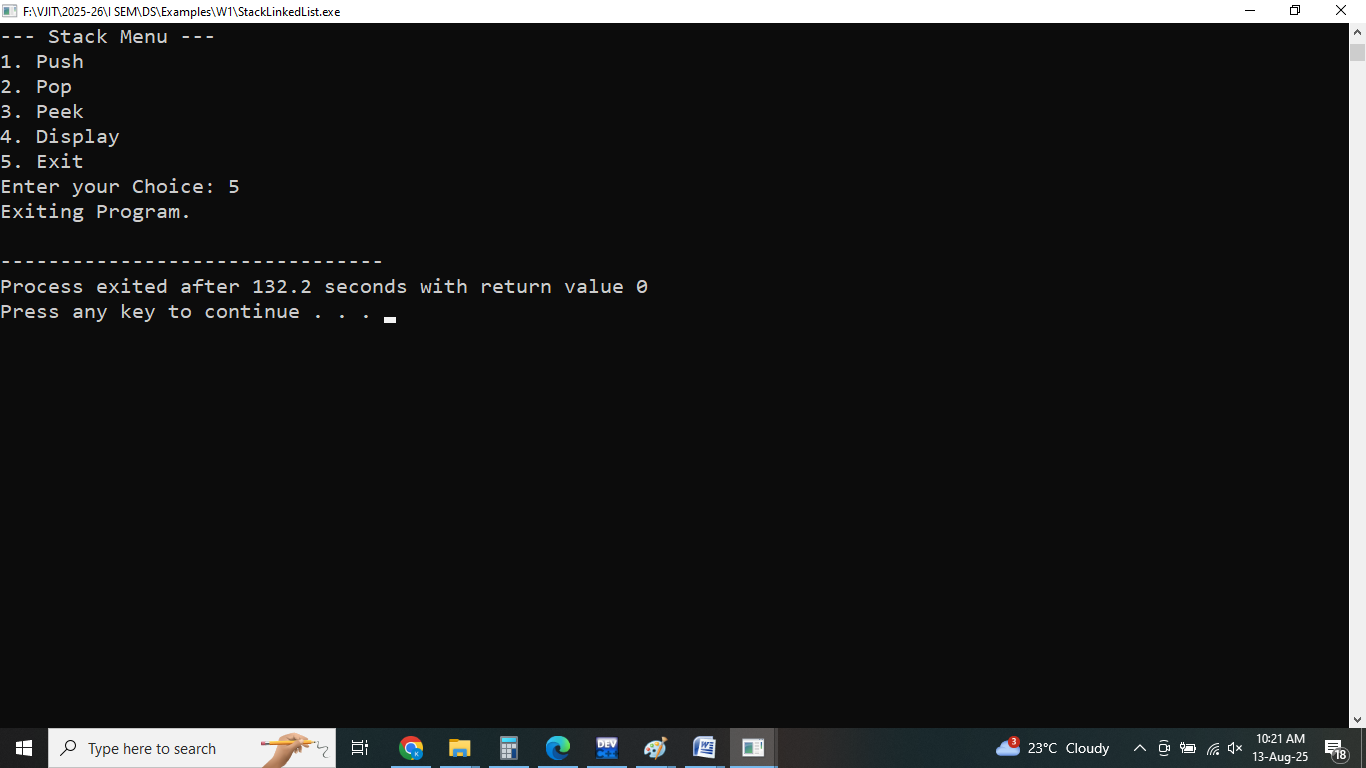
return 0;

}

**Output:**

****

****

****

1. **C Programs to implement Queue Operations using Linked List**

#include <stdio.h>

#include <stdlib.h>

*// Define structure for a queue node*

struct **Node**

{

int data;

struct Node\* next;

};

*// Initialize front and rear pointers*

struct Node\* front = NULL;

struct Node\* rear = NULL;

*// Enqueue operation*

void **enqueue**(int value)

{

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (!newNode)

{

printf("Heap overflow\n");

return;

}

newNode->data = value;

newNode->next = NULL;

if (rear == NULL)

{

*// Queue is empty*

front = rear = newNode;

} else {

rear->next = newNode;

rear = newNode;

}

printf("Enqueued %d into Queue\n", value);

}

*// Dequeue operation*

void **dequeue**()

{

if (front == NULL)

{

printf("Queue underflow\n");

return;

}

struct Node\* temp = front;

printf("Dequeued %d from Queue\n", front->data);

front = front->next;

if (front == NULL)

{

rear = NULL;

}

free(temp);

}

*// Peek operation*

void **peek**()

{

if (front == NULL)

{

printf("Queue is empty\n");

return;

}

printf("Front element is %d\n", front->data);

}

*// Display queue*

void **display**()

{

struct Node\* temp = front;

if (temp == NULL)

{

printf("Queue is empty\n");

return;

}

printf("Queue elements: ");

while (temp != NULL)

{

printf("%d <- ", temp->data);

temp = temp->next;

}

printf("NULL\n");

}

*// Main function to test queue operations*

int **main**()

{

int choice, value;

while (1)

{

printf("\n--- Queue Menu ---\n");

printf("1. Enqueue\n2. Dequeue\n3. Peek\n4. Display\n5. Exit\n");

printf("Enter your Choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

printf("Enter value to Enqueue: ");

scanf("%d", &value);

enqueue(value);

break;

case 2:

dequeue();

break;

case 3:

peek();

break;

case 4:

display();

break;

case 5:

printf("Exit from the Program\n");

exit(0);

default:

printf("Invalid Choice\n");

}

}

return 0;

}

**Output:**

