**LAB 6**

**Name : Abhinav Sanjay**

**USN : 1BM23CS009**

1. **Implement queue using linked list**

#include <stdio.h>

#include <stdlib.h>

struct node {

int value;

struct node \*next;

};

typedef struct node\* NODE;

NODE getnode() {

NODE new\_node = (NODE)malloc(sizeof(struct node));

if (new\_node == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

return new\_node;

}

void display(NODE first) {

NODE temp;

if (first == NULL) {

printf("Linked list is empty.\n");

return;

}

temp = first;

while (temp != NULL) {

printf("%d ", temp->value); // Improved readability with space between nodes

temp = temp->next;

}

printf("\n"); // End of list indication

}

NODE insert\_beg(int item, NODE first) {

NODE new = getnode();

new->value = item;

new->next = first;

return new; // return the new head of the list

}

NODE delete\_end(NODE first) {

if (first == NULL) {

printf("Linked list is empty.\n");

return NULL;

}

if (first->next == NULL) { // Only one element

printf("Deleted element: %d\n", first->value); // Print deleted value

free(first);

return NULL;

}

NODE prev = NULL, last = first;

while (last->next != NULL) {

prev = last;

last = last->next;

}

printf("Deleted element: %d\n", last->value); // Print deleted value

prev->next = NULL;

free(last);

return first; // Return the updated list

}

int main() {

int choice, item;

NODE first = NULL, item\_del; // Initialize first to NULL to represent an empty list

while (1) {

printf("Enter your choice\n1. Insert\n2. Delete\n3. Display\n4.Exit\n");

scanf("%d", &choice);

switch (choice) {

case 1:

// Insert item at the beginning

printf("Enter item to insert: ");

scanf("%d", &item);

first = insert\_beg(item, first); // Update first with the new head

break;

case 2:

// Delete item from the end

first = delete\_end(first); // Update first with the new head after deletion

break;

case 3:

// Display the linked list

printf("The list is being displayed:\n");

display(first);

break;

default:

printf("Exiting...\n");

exit(0); // Exit if the user enters an invalid choice

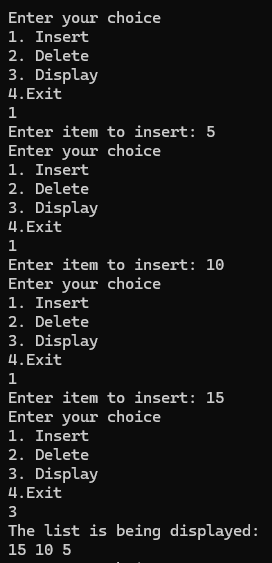
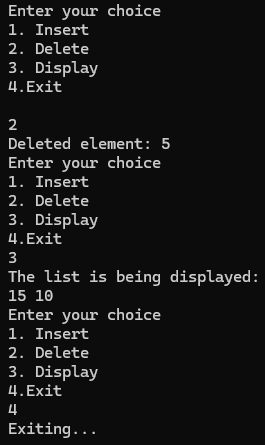
}

}

return 0; // Return 0 to indicate successful execution

}

**Output:**

** **

1. **Implement stack using linked list**

#include <stdio.h>

#include <stdlib.h>

struct node {

int value;

struct node \*next;

};

typedef struct node\* NODE;

NODE getnode() {

NODE new\_node = (NODE)malloc(sizeof(struct node));

if (new\_node == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

return new\_node;

}

void display(NODE first) {

NODE temp;

if (first == NULL) {

printf("Linked list is empty\n");

return;

}

temp = first;

while (temp != NULL) {

printf("%d ", temp->value);

temp = temp->next;

}

printf("\n");

}

NODE insert\_beg(int item, NODE first) {

NODE new = getnode();

new->value = item;

new->next = first;

return new; // Return the new head of the stack

}

NODE delete\_first(NODE first, int \*item) {

if (first == NULL) {

printf("Stack is empty\n");

return NULL;

}

NODE temp = first;

\*item = temp->value; // Store the deleted value

first = first->next;

free(temp); // Free the memory of the deleted node

return first;

}

int main() {

int choice, item;

NODE first = NULL; // Initialize first as NULL (empty stack)

while (1) {

printf("Enter your choice\n 1.Push\n 2.Pop\n 3.Display\n 4.Exit\n");

scanf("%d", &choice);

switch (choice) {

case 1: // Push operation

printf("Enter item to push:\n");

scanf("%d", &item);

first = insert\_beg(item, first); // Capture the return value to update 'first'

break;

case 2: // Pop operation

first = delete\_first(first, &item); // Capture the updated first

if (first != NULL) {

printf("Item popped: %d\n", item);

}

break;

case 3: // Display stack

printf("The stack is being displayed: ");

display(first);

break;

case 4: // Exit

printf("Exiting...\n");

exit(0);

default:

printf("Invalid choice. Try again.\n");

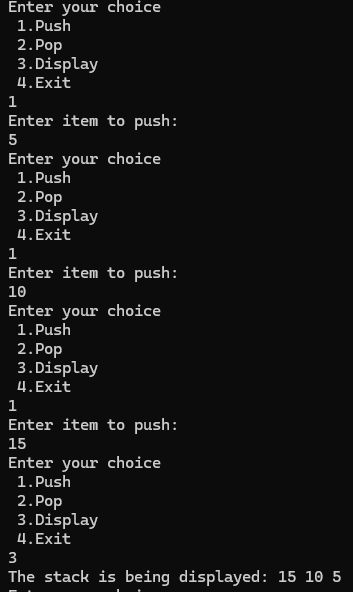
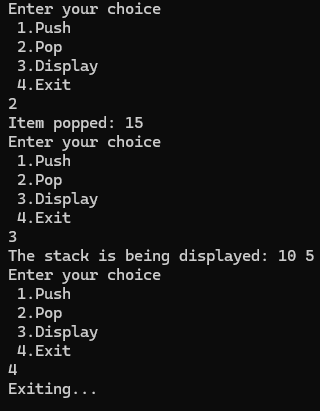
}

}

return 0;

}

**Output:**

** **