5/01/24

Stack Implementation with linked list

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

struct Node {
        int data
     struct Node* next;
};

struct Node* createNode(int data){
     struct Node* newNode =(struct Node*)malloc(
     Size of(struct Node));
     newNode->data =data;
     newNode->next = NULL;
      return newNode;
  }
void push(struct Node* top, int data) {
     struct Node* newNode = createNode(data);
     newNode->next = *Top;
     * Top =newNode;
  }

~~inp~~ int pop(struct Node** top) {
     if(*Top ==NULL) {
      printf("stack is empty.\n");
```

28/1/24

```
    return -1;
}
int data = (* top)->data;
struct node * temp = * top;
* top = (* top)->next;
free(temp)
return data;
}
void printstack(struct Node* top) {
    printf("stack elements:");
    while (top)->next;
    free(temp);
    return data;
}
    printf("\n");
}
int main() {
        struct Node* stackTop = NULL";
        int choice, data;
        do {
            printf("\n choose an operator:\n");
            printf(" 1. push\n");
            printf(" 2. pop \n");
            printf(" 3. print stack\n");
            scanf("%d", &choice);
```

```
switch (choice) {
  case 1:
      printf ("Enter the element to push into the stack:");
      scanf("%d", &data);
      push(&stackTop, data);
      break;

  case 2:
     printf(" Popped element from the stack: %d\n",
     pop(&stackTop));
         break;

  case 3:
      printStack(stackTop);
        break;

  case 0:
     printf("Exiting program.\n");
      break;

  default:
     printf("Invalid choice. please enter a valid
      option.\n");
     }
} while(choice != 0);
 return 0
}
```

Output :-

```
choose an operation:
1. Enqueue
2. Dequeue
3. Print Queue
0. Exit

Enter the element to equeue into the queue :22
Enter the element to equeue into the queue :23
Enter the element to equeue into the queue :24

Enter your choice :2
Dequeued elements from the queue :22

Enter your choice :3
Queue elements :22 23 24
```