reverse.c X | mergesort.c X | concatination.c X | search.c X | *stack.c X | *queue.c X

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

void push();
void pop();
void display();
struct node
{
    int data;
    struct node *next;
};
struct node *top=NULL;

int main(int argc, char **argv)
{
    int choice;
    char ch;
    do
    {
    printf("\n1. Push \n2. Display  \n3. Pop\n");
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: push(); break;
        case 2: display();break;
        case 3: pop(); break;
    }
    printf("\nDo you want to continue (y||Y):");
    fflush(stdin);
    scanf("%c",&ch);
    }while(ch=='y'||ch=='Y');
}

void push()
```

```
}

void push()
{
    int item;
    struct node *newnode;
    printf("Enter the element\n");
    scanf("%d",&item);

    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=item;
    newnode->next=NULL;
    if(top==NULL)
        top=newnode;
    else
        newnode->next=top;
        top=newnode;
}
void pop()
{
    if(top==NULL)
        printf("stack is empty");
    else
    {

        printf("element removed is %d:", top->data);

        top=top->next;

    }

}

void display()
```

```
        top=newnode;
    else
        newnode->next=top;
        top=newnode;
}
void pop()
{
    if(top==NULL)
        printf("stack is empty");
    else
    {
        printf("element removed is %d:", top->data);
        top=top->next;
    }

}

void display()
{
struct node *temp;
temp=top;
if(top==NULL)
    printf("Stack is empty");
while(temp!=NULL)
{
    printf("%d",temp->data);
    temp=temp->next;
}

}
```