**29/1/2024 lab programs**

**Single linked list : sort , reverse , concatenation**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head=NULL;

void create()

{

struct node \*new\_node,\*ptr;

int num;

printf("\n enter -1 to end\n");

printf("enter data: \n");

scanf("%d",&num);

while(num!=-1)

{

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

new\_node->data=num;

if(head==NULL)

{

new\_node->next=NULL;

head=new\_node;

}

else

{

ptr=head;

while(ptr->next!=NULL)

ptr=ptr->next;

new\_node->next=NULL;

ptr->next=new\_node;

}

printf("enter data:");

scanf("%d",&num);

}

}

void display()

{

struct node \*ptr;

ptr=head;

while(ptr!=NULL)

{

printf("%d \n",ptr->data);

ptr=ptr->next;

}

}

void sort()

{

struct node \*ptr1,\*ptr2;

int temp;

ptr1=head;

while(ptr1->next!=NULL)

{

ptr2=ptr1->next;

while(ptr2!=NULL)

{

if(ptr1->data > ptr2->data)

{

temp=ptr1->data;

ptr1->data=ptr2->data;

ptr2->data=temp;

}

ptr2=ptr2->next;

}

ptr1=ptr1->next;

}

}

void con()

{

struct node \*new\_node,\*h1,\*h2,\*ptr;

int i,n,m;

printf("enter no. of elements in 1st list:\n");

scanf("%d",&n);

h1=NULL;

for(i=0;i<n;i++)

{

printf("enter data:\n");

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

scanf("%d",&new\_node->data);

new\_node->next=NULL;

if(h1==NULL)

{

h1=new\_node;

}

else

{

ptr=h1;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=new\_node;

}

}

printf("enter no. of elements in 2nd list:\n");

scanf("%d",&m);

h2=NULL;

for(i=0;i<m;i++)

{

printf("enter data:\n");

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

scanf("%d",&new\_node->data);

new\_node->next=NULL;

if(h2==NULL)

{

h2=new\_node;

}

else

{

ptr=h2;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=new\_node;

}

}

concatenate(h1,h2);

}

void concatenate(struct node \*h1,struct node \*h2)

{

struct node \*ptr;

head=h1;

ptr=head;

if(h1==NULL && h2==NULL)

{

printf("list is empty!");

}

else

{

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=h2;

display();

}

}

void reverse()

{

struct node \*current , \*ptr , \*temp;

ptr=NULL;

current=head;

while(current!=NULL)

{

temp=current->next;

current->next=ptr;

ptr=current;

current=temp;

}

head=ptr;

display();

}

void main()

{

int choice;

while(1)

{

printf("\n Enter \n 1:create \n 2:sort \n 3:concatenate \n 4:reverse \n 5:exit");

scanf("%d",&choice);

switch(choice)

{

case 1:

create();

break;

case 2:

sort();

display();

break;

case 3:

con();

break;

case 4:

reverse();

break;

case 5:

exit(0);

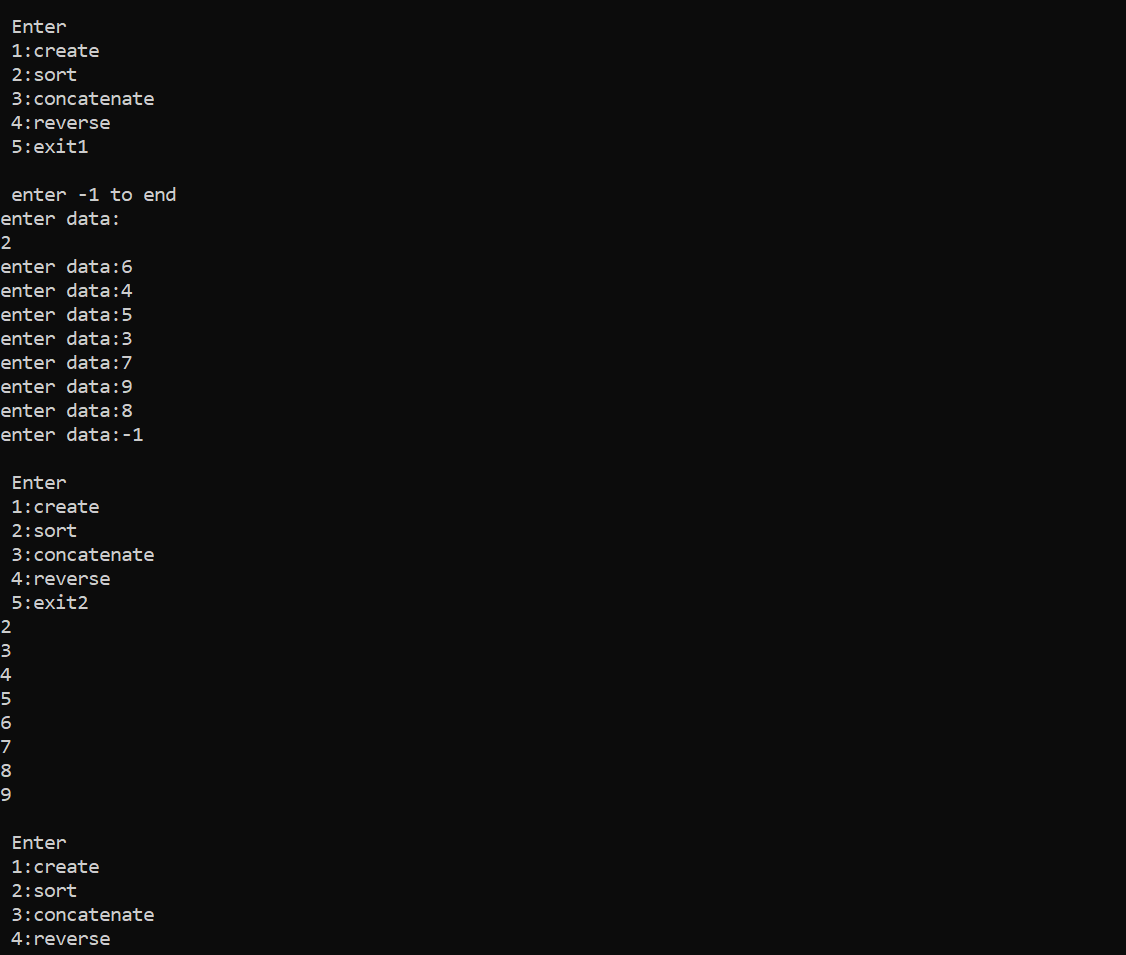
default:

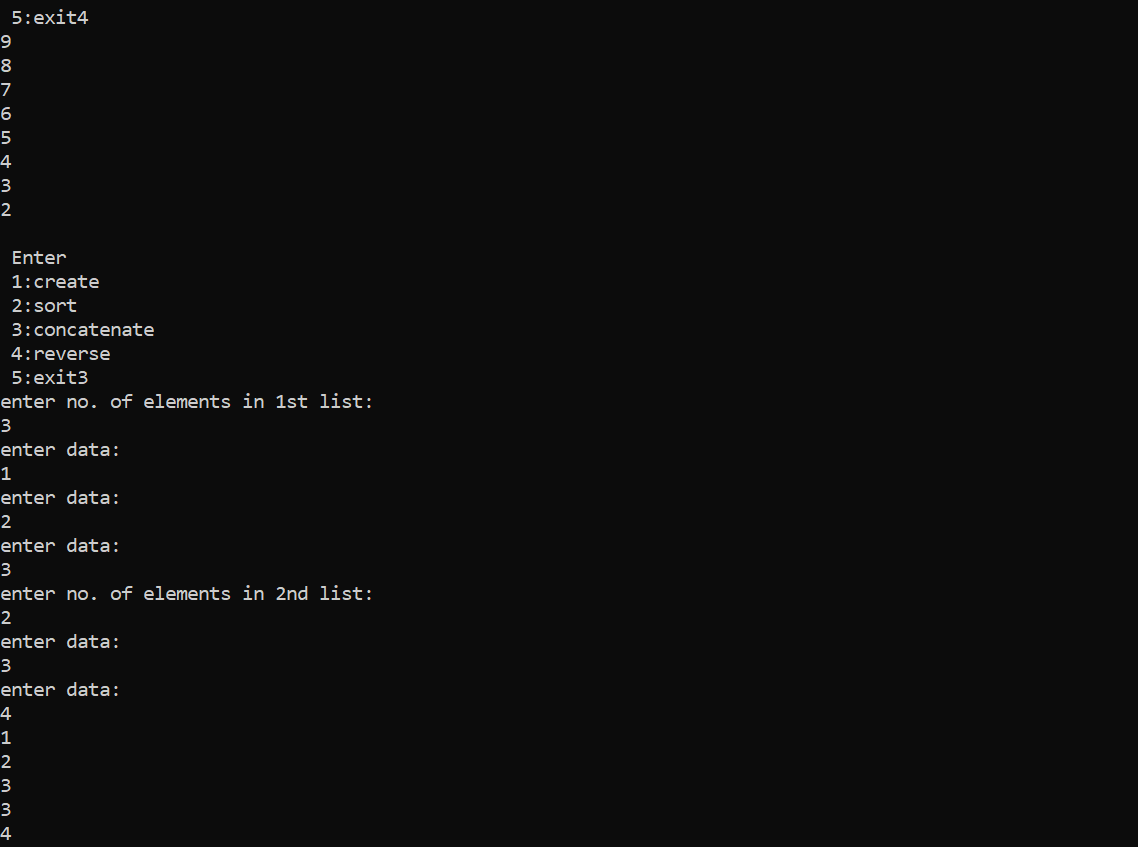
printf("invalid input");

}

}

}





**Stack Implementation using single linked list:**

#include <stdio.h>

#include <stdlib.h>

void push();

void pop();

void display();

struct node

{

int data;

struct node \*next;

};

struct node\* top=NULL;

void main()

{

int choice;

while(1)

{

printf("\n enter \n 1:push \n 2:pop \n 3:display 4:exit");

scanf("%d",&choice);

switch(choice)

{

case 1:

push();

printf("\n the stack is:\n");

display();

break;

case 2:

pop();

printf("\n the stack is:\n");

display();

break;

case 3:

printf("\n the stack is:\n");

display();

break;

case 4:

exit(0);

default:

printf("Invalid Input!");

}

}

}

void push()

{

struct node \*temp;

int val;

printf("enter the data to be pushed:");

scanf("%d",&val);

if(top==NULL)

{

top=malloc(sizeof(struct node));

top->data=val;

top->next=NULL;

}

else

{

temp=malloc(sizeof(struct node));

temp->data=val;

temp->next=top;

top=temp;

}

}

void pop()

{

struct node \*temp;

if(top==NULL)

{

printf("Stack is Empty!");

}

else

{

temp=top;

top=top->next;

printf("\n the data popped is %d ",temp->data);

free(temp);

}

}

void display()

{

struct node \*temp;

temp=top;

if(top==NULL)

{

printf("stack is empty!");

}

else

{

while(temp!=NULL)

{

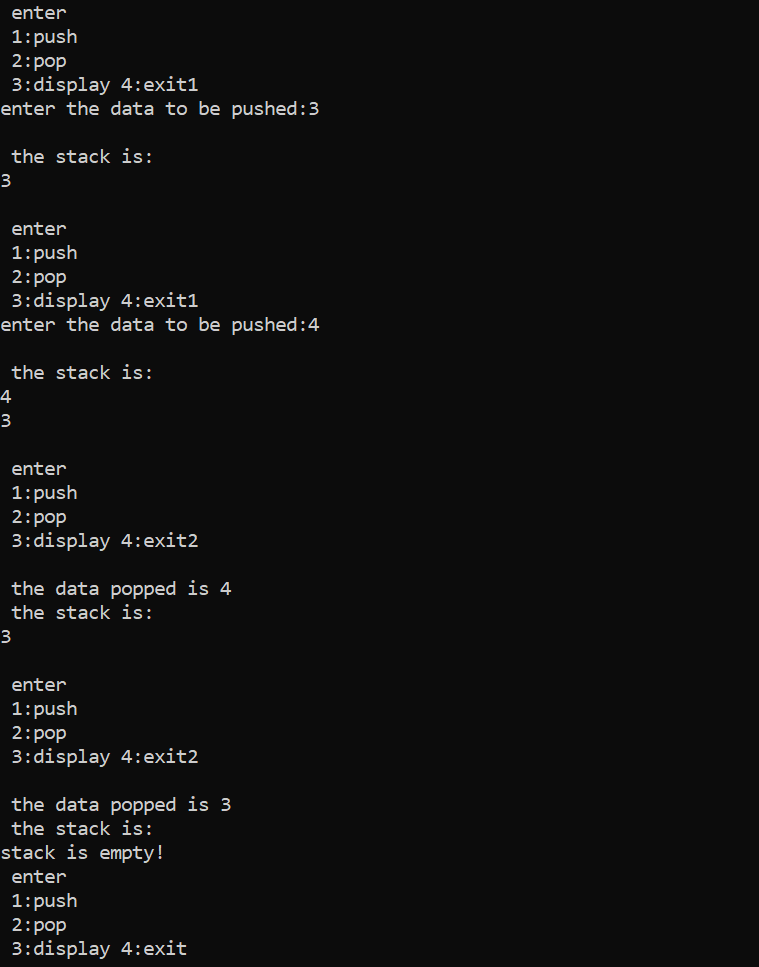
printf("%d\n",temp->data);

temp=temp->next;

}

}

}



**Queue Implementation using linked list:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*front=NULL , \*rear=NULL;

void enqueue();

void dequeue();

void display();

int main()

{

int choice;

while(1)

{

printf("\n Enter \n 1:enqueue \n 2:dequeue \n 3:display \n 4:exit \n");

scanf("%d",&choice);

switch(choice)

{

case 1:

enqueue();

printf("the queue is:");

display();

break;

case 2:

dequeue();

printf("\n the queue is:");

display();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

printf("Invalid Input!");

}

}

return 0;

}

void enqueue()

{

int val;

printf("enter data to be inserted:\n");

scanf("%d",&val);

struct node \*temp;

temp=(struct node \*)malloc(sizeof(struct node));

temp->data=val;

temp->next=NULL;

if(front==NULL && rear==NULL)

{

front=rear=temp;

}

else

{

rear->next=temp;

rear=temp;

}

}

void dequeue()

{

struct node \*temp;

temp=front;

front=front->next;

printf("the removed data is: %d ",temp->data);

free(temp);

}

void display()

{

struct node \*temp;

temp=front;

if(front==NULL)

{

printf("Queue is empty!");

}

else

{

while(temp!=NULL)

{

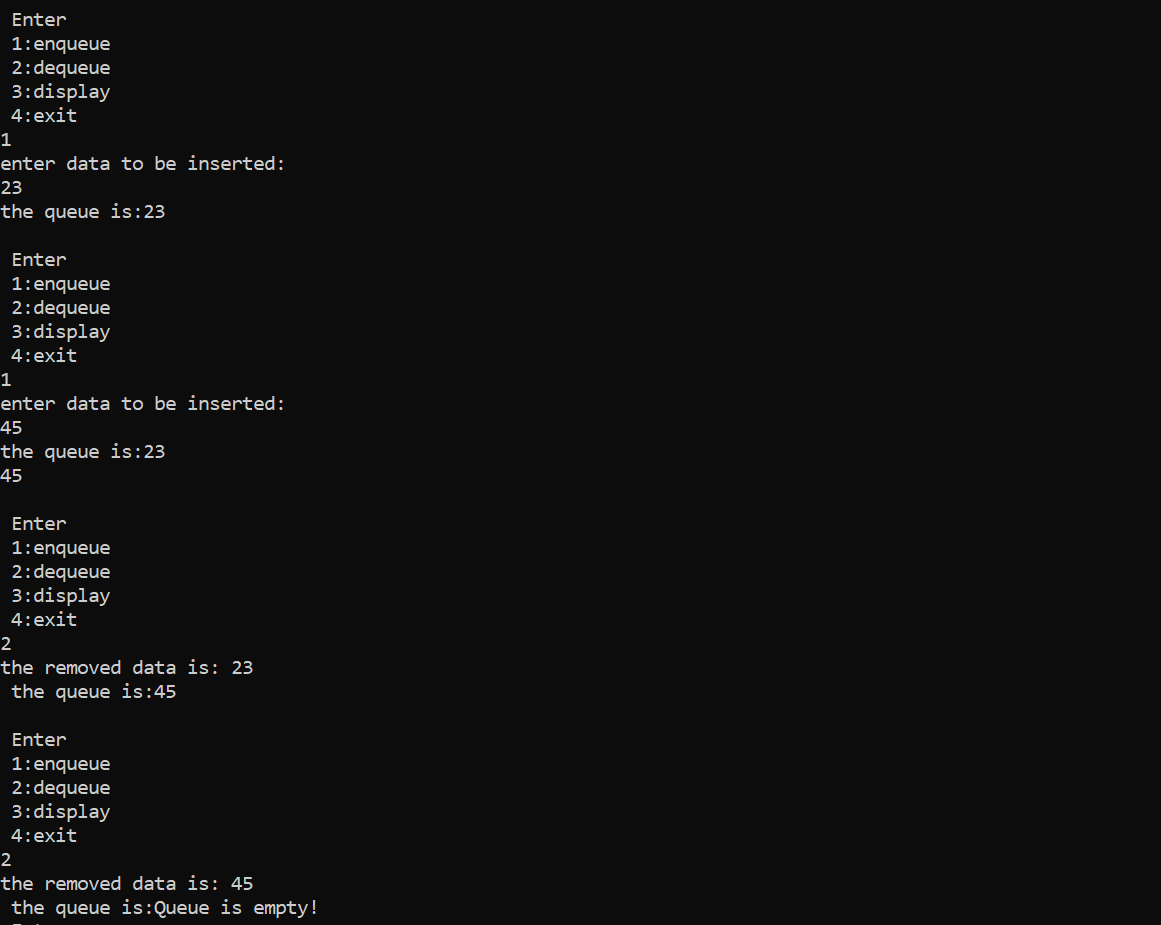
printf("%d \n",temp->data);

temp=temp->next;

}

}

}



**Observation book:**

