DS LAB

CYCLE1\_CO1

SUBMITTED BY:

SONI.R

MCA:134

Stack:

#include<stdio.h>

#include<conio.h>

int stack[100];

int i,j,n;

int choice=0;

int top=-1;

int push();

int pop();

int show();

int topp();

int main()

{

printf("enter the elements in stack:");

scanf("%d",&n);

while(choice != 4)

{

printf("choose the stack operation to be performed");

printf("\n1.Push\n2.Pop\n3.show\n4.Top\n");

printf("enter Your choice:\n");

scanf("%d",&choice);

switch (choice)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

show();

break;

case 4:

topp();

break;

default:

printf("Invalid entry");

break;

};

}

return 0;

}

int push()

{

int val;

if(top==n)

{

printf("\n overflow");

}

else

{

printf("enter the value:");

scanf("%d",&val);

top=top+1;

stack[top]=val;

}

}

int pop()

{

if(top==-1)

{

printf("\n underflow");

}

else

{

top=top-1;

}

}

int show()

{

if (top==-1)

{

printf("stack is empty");

}

else

{

for ( i = top; i >=0; i--)

{

printf("%d\n",stack[i]);

}

}

}

int topp()

{

if(top==-1)

{

printf("stack is empty");

}

else

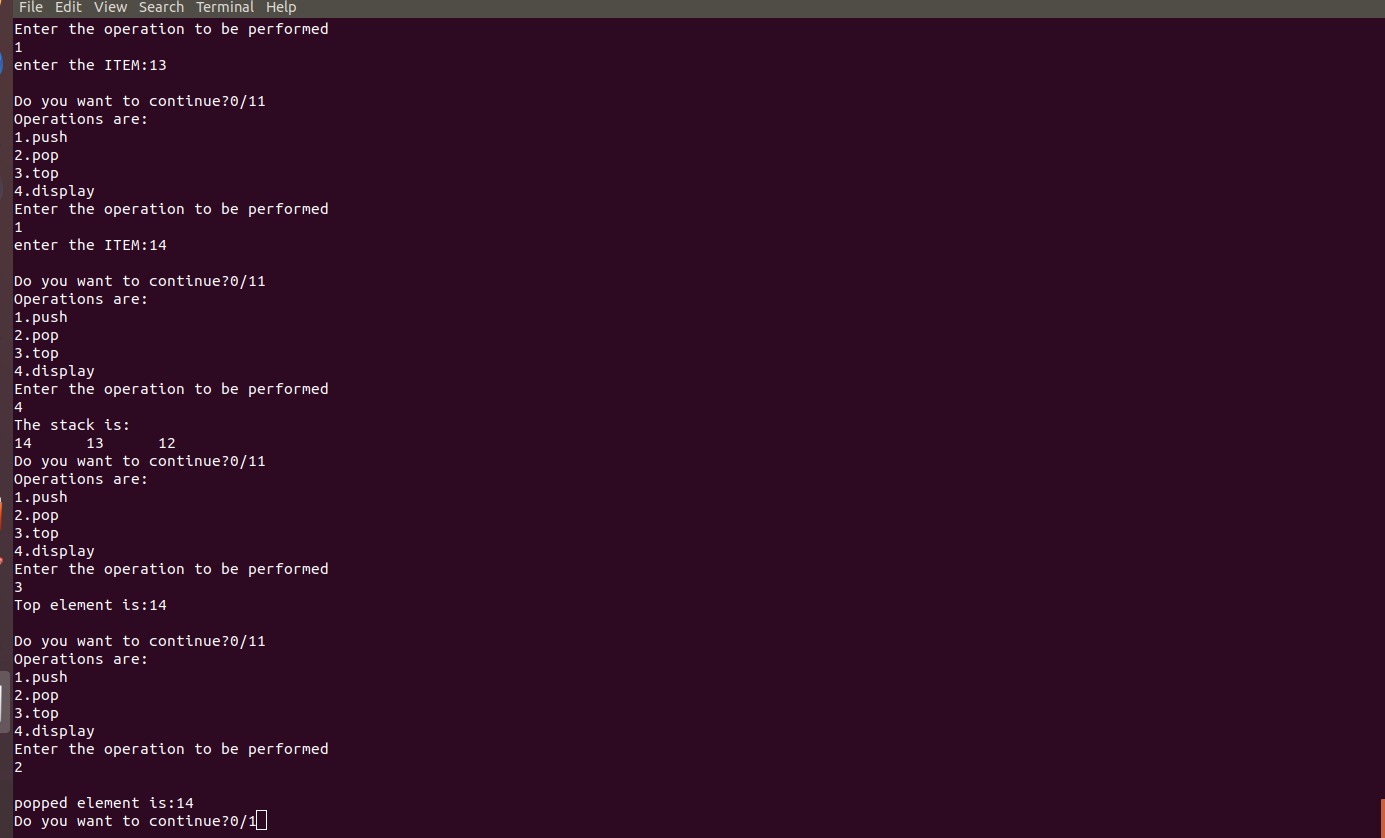
{

printf("%d",stack[top]);

}

}

Output:



Queue:

#include<stdio.h>

#include<stdlib.h>

#define maxsize 5

int insert();

int delete();

int display();

int peek();

int front=-1,rear=-1;

int queue[maxsize];

int insert()

{

int item;

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

scanf("%d",&item);

if(rear==maxsize-1)

{

printf("\n Overflow");

return 0;

}

if(front==-1&&rear==-1)

{

front=0;

rear=0;

}

else

{

rear=rear+1;

}

queue[rear]=item;

printf("\nValue inserted");

}

int delete()

{

int item;

if(front==-1||front>rear)

{

printf("undereflow");

return 0;

}

else

{

item=queue[front];

if(front==rear)

{

front=-1;

rear=-1;

}

else

{

front=front+1;

}

printf("%d is deleted is ",item);

}

}

int display()

{

int i;

if(rear==-1)

{

printf("\nEmpty Queue");

}

else

{

printf("\n The Queue elements are:");

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

{

printf("\n%d",queue[i]);

}

}

}

int peek()

{

if(front==-1||front>rear)

{

printf("\n Queue is empty");

return -1;

}

else

{

printf("peek value is %d",queue[front]);

}

return 0;

}

int main()

{

int choice;

while(choice!=5)

{

printf("\nChoose an operation\n");

printf("\n1.Insert an element\n2.Delete an element\n3.Display the elemenets of queue\n4 peek Value\n5.Exit\n");

printf("enter your choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

peek();

break;

case 5:

exit(0);

break;

default:

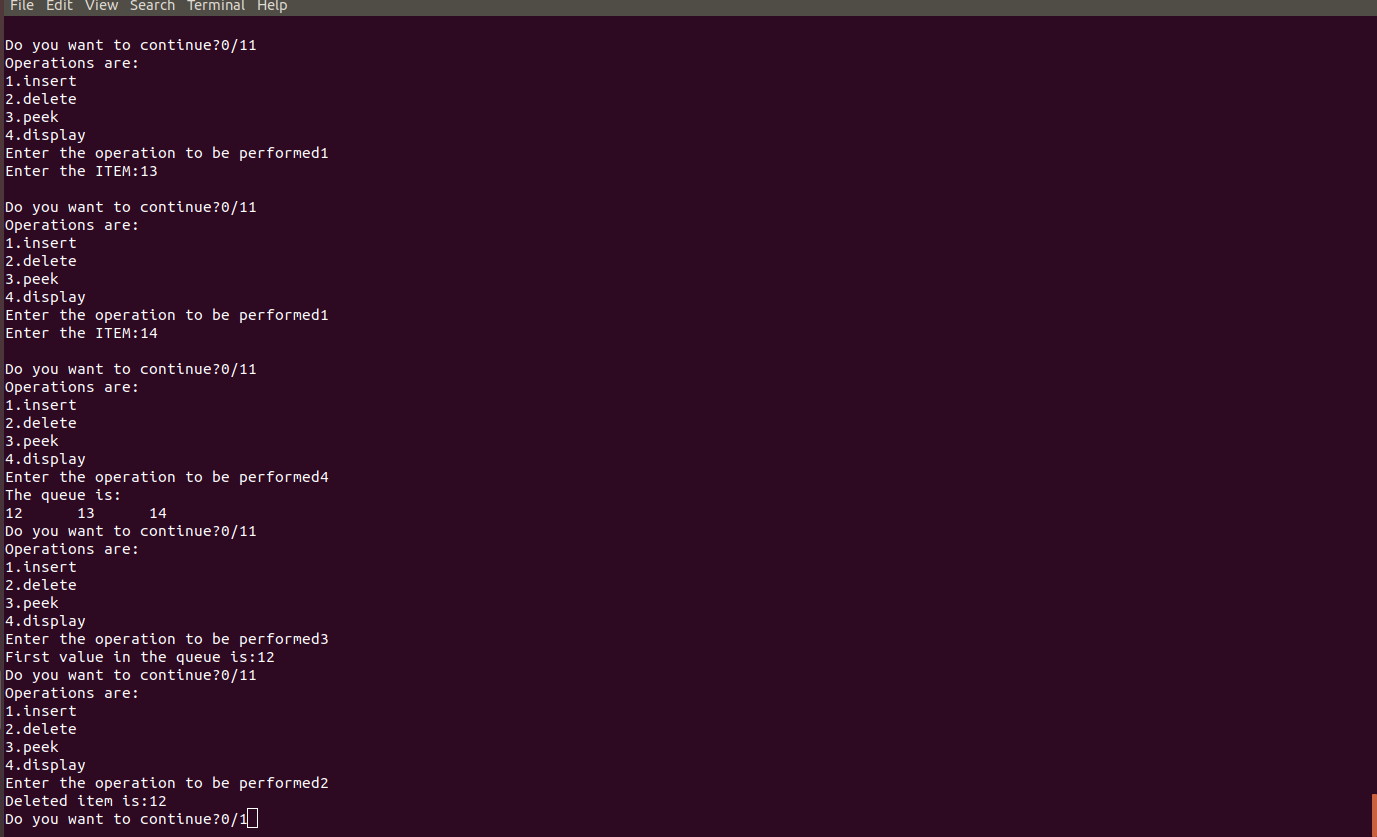
printf("\nEnter Valid choice:\n");

}

}

}

Output:



Singly linkedlist:

#include<stdio.h>

#include<stdlib.h>

int beginsert();

int begdelete();

int lastinsert();

int traverse();

int posinsert();

int display();

int search();

int posdelete();

struct node

{

int data;

struct node \*link;

};

struct node \*head;

int beginsert()

{

struct node \*ptr;

int item;

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

if(ptr==NULL)

{

printf("\n List is empty");

}

else

{

printf("\nEnter the element to be inserted:");

scanf("%d",&item);

ptr->data=item;

ptr->link=head;

head=ptr;

printf("%d is inserted at the beginning",item);

}

}

int begdelete()

{

struct node \*ptr;

if(head==NULL)

printf("The list is empty");

else

{

ptr = head;

head = ptr->link;

free(ptr);

printf("\nNode from beginning is deleted\n");

}

}

int traverse()

{

int count=0;

struct node \*ptr=NULL;

ptr=head;

if(head==NULL)

{

printf("list is empty");

}

else{

while(ptr!=NULL)

{

ptr=ptr->link;

count++;

}

printf("\nThe total Number of elements are %d",count);

}

}

int lastinsert()

{

struct node \*ptr,\*temp;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value?\n");

scanf("%d",&item);

ptr->data = item;

if(head == NULL)

{

ptr -> link = NULL;

head = ptr;

printf("\nNode inserted");

}

else

{

temp = head;

while (temp -> link!= NULL)

{

temp = temp -> link;

}

temp->link = ptr;

ptr->link = NULL;

printf("\nNode inserted");

}

}

}

int lastdelete()

{

struct node \*ptr,\*ptr1;

if(head == NULL)

{

printf("\nlist is empty");

}

else if(head -> link == NULL)

{

head = NULL;

free(head);

printf("\n Node deleted\n");

}

else

{

ptr = head;

while(ptr->link != NULL)

{

ptr1 = ptr;

ptr = ptr ->link;

}

ptr1->link = NULL;

free(ptr);

printf("\nNode deleted\n");

}

}

int posinsert()

{

int loc,i,item;

struct node \*ptr,\*temp;

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

if(ptr==NULL)

{

printf("UNDERFLOW");

}

printf("\nEnter the location:\n");

scanf("%d",&loc);

temp=head;

for(i=1;i<loc-1;i++)

{

temp=temp->link;

if(temp==NULL)

{

printf("\nThere is less than %d elements\n",loc);

}

}

printf("enter the data:");

scanf("%d",&item);

ptr->data=item;

ptr->link=temp->link;

temp->link=ptr;

printf("\nNode Inserted\n");

}

int search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

printf("\nEnter item to search:\n");

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("\nitem found at location %d ",i+1);

flag=0;

break;

}

else

{

flag=1;

}

i++;

ptr = ptr -> link;

}

if(flag==1)

{

printf("\nItem not found\n");

}

}

}

int display()

{

struct node \*ptr;

ptr=head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

while(ptr!=NULL)

{

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

ptr=ptr->link;

}

}

}

int posdelete()

{

int i,pos;

struct node \*ptr;

printf("enter the position:");

scanf("%d",&pos);

ptr=head;

for(i=0;i<pos-1;i++)

{

ptr=ptr->link;

}

ptr->link=ptr->link->link;

printf("the deleted element is %d",ptr->data);

}

void main()

{

int choice =0;

while(choice != 30)

{

printf("\n1.Insert in begining\n2.Delete from beggining\n3.Traverse\n4.Delete from beggining\n5.last insert\n6.last delete\n7.insert specific position\n8.display\n9.search\n10.pos delete");

printf("\nEnter your choice?\n\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

beginsert();

break;

case 2:

begdelete();

break;

case 3:

traverse();

break;

case 4: begdelete();

break;

case 5:

lastinsert();

break;

case 6:

lastdelete();

break;

case 7:

posinsert();

break;

case 8:

display();

break;

case 9:

search();

break;

case 10:

posdelete();

break;

default:

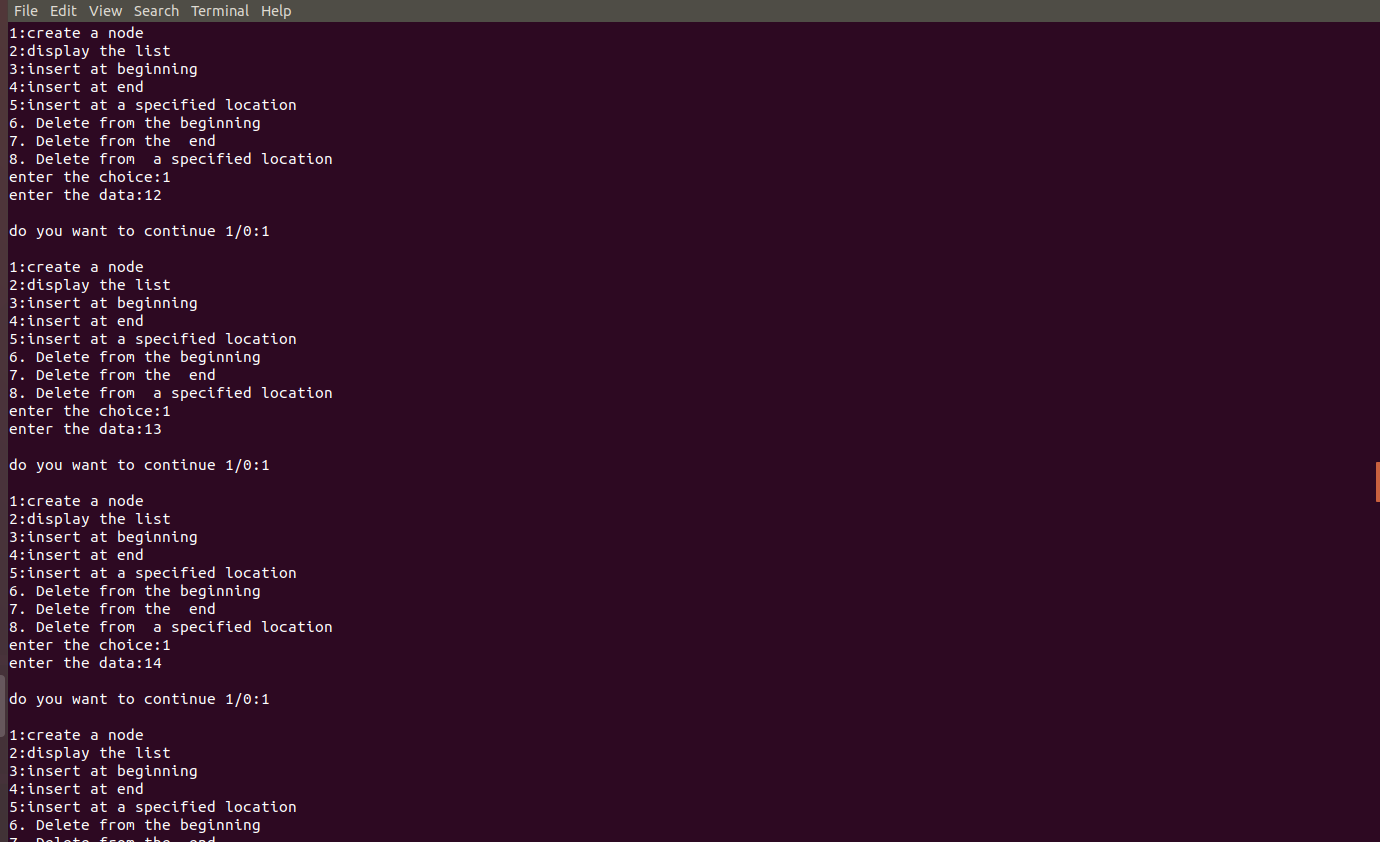
printf("invalid entry");

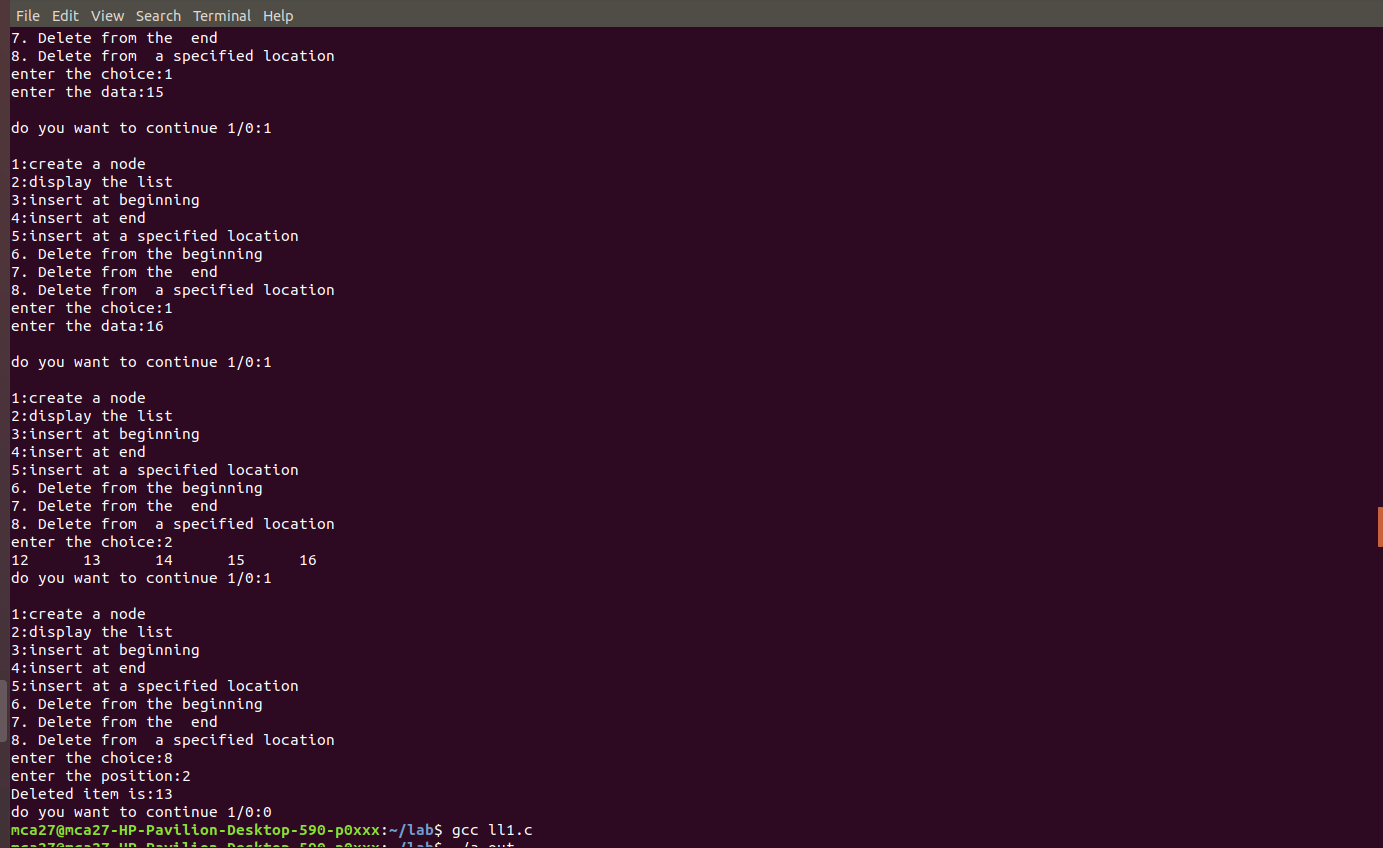
}

}

}

Output:

****

****

****

Doubly linkedlist:

#include<stdio.h>

#include<stdlib.h>

void insertbeg();

void display();

void insertend();

void deletebeg();

void deleteend();

void posinsert();

void posdelete();

void search();

struct node

{

int data;

struct node \*next;

struct node \*prev;

};

struct node \*head;

void insertbeg()

{

int item;

struct node \*ptr;

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

if(ptr==NULL)

{

printf("\noverflow\n");

}

else

{

printf("\nEnter the data:\n");

scanf("%d",&item);

if(head==NULL)

{

ptr->next=NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else

{

ptr->data=item;

ptr->prev=NULL;

ptr->next=head;

head->prev=ptr;

head=ptr;

}

printf("Node Inserted");

}

}

void display()

{

struct node \*ptr;

ptr=head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void insertend()

{

int item;

struct node \*ptr,\*temp;

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

if(ptr==NULL)

{

printf("overflow");

}

else

{

printf("enter the data:");

scanf("%d",&item);

ptr->data=item;

if(head==NULL)

{

ptr->next=NULL;

ptr->prev=NULL;

head=ptr;

}

else

{

temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=ptr;

ptr->next=NULL;

ptr->prev=temp;

}

}

}

void deletebeg()

{

struct node \*ptr;

if(head==NULL)

{

printf("Underflow");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("\nNode deleted\n");

}

else

{

ptr=head;

head=head->next;

free(head->prev);

head->prev=NULL;

printf("\nNode deleted\n");

}

}

void deleteend()

{

struct node \*ptr;

if(head==NULL)

{

printf("overflow");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("\nNode deleted\n");

}

else

{

ptr=head;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->prev->next=NULL;

free(ptr);

printf("\nNode deleted\n");

}

}

void posinsert()

{

int loc,i,item;

struct node \*ptr,\*temp;

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

if(ptr==NULL)

{

printf("UNDERFLOW");

}

printf("\nEnter the location:\n");

scanf("%d",&loc);

if (loc==1)

{

printf("\nenter the data:\n");

scanf("%d",&item);

ptr->data=item;

ptr->prev=NULL;

ptr->next=head;

head->prev=ptr;

head=ptr;

}

else

{

temp=head;

for(i=1;i<loc-1;i++)

{

temp=temp->next;

if(temp==NULL)

{

printf("\nThere is less than %d elements\n",loc);

}

}

printf("enter the data:");

scanf("%d",&item);

ptr->data=item;

ptr->next=temp->next;

ptr->prev=temp;

temp->next=ptr;

temp->next->prev=ptr;

printf("\nNode Inserted\n");

}

}

void posdelete()

{

struct node \*ptr;

int loc,i;

printf("enter the position:");

scanf("%d",&loc);

if (head == NULL || loc<=0)

printf("empty list");

ptr=head;

for (int i = 1; ptr != NULL && i < loc; i++)

ptr = ptr->next;

if (ptr == NULL)

printf("Not possible");

if (head == NULL || ptr == NULL)

printf("Not possible");

if (head== ptr)

head = ptr->next;

if (ptr->next != NULL)

ptr->next->prev = ptr->prev;

if (ptr->prev != NULL)

ptr->prev->next = ptr->next;

free(ptr);

}

void search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

printf("\nEnter item to search:\n");

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("\nitem found at location %d ",i+1);

flag=0;

break;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

if(flag==1)

{

printf("\nItem not found\n");

}

}

}

void main()

{

int choice =0;

while(choice != 30)

{

printf("\n1.Insert in begining\n2.Display\n3.Insert at end\n4.Delete from beggining\n5.delete from end\n6.Insert at specified postion\n7.Delete from a given position\n8.Search");

printf("\nEnter your choice?\n\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1:

insertbeg();

break;

case 2:

display();

break;

case 3:

insertend();

break;

case 4:

deletebeg();

break;

case 5:

deleteend();

break;

case 6:

posinsert();

break;

case 7:

posdelete();

break;

case 8:

search();

break;

default:

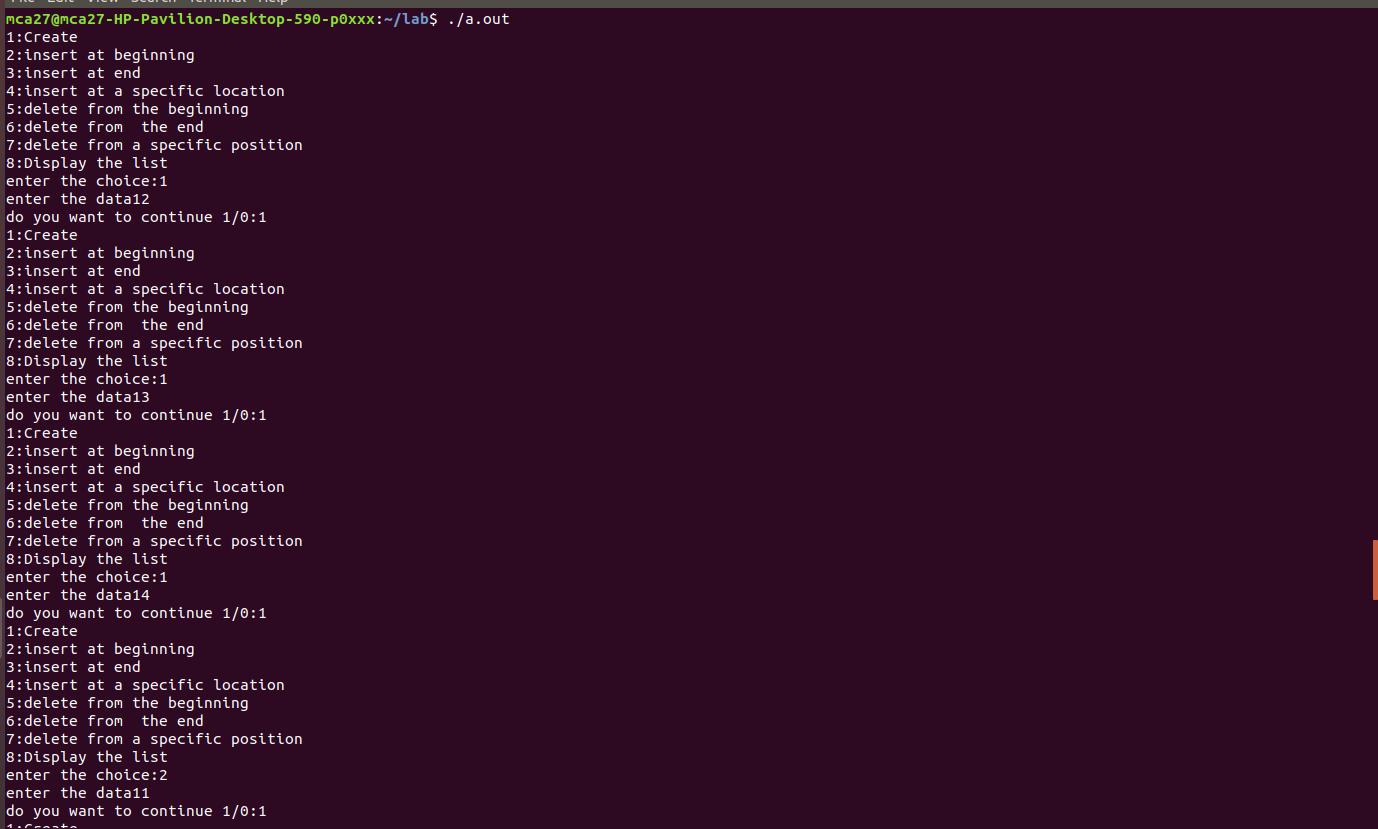
printf("invalid entry");

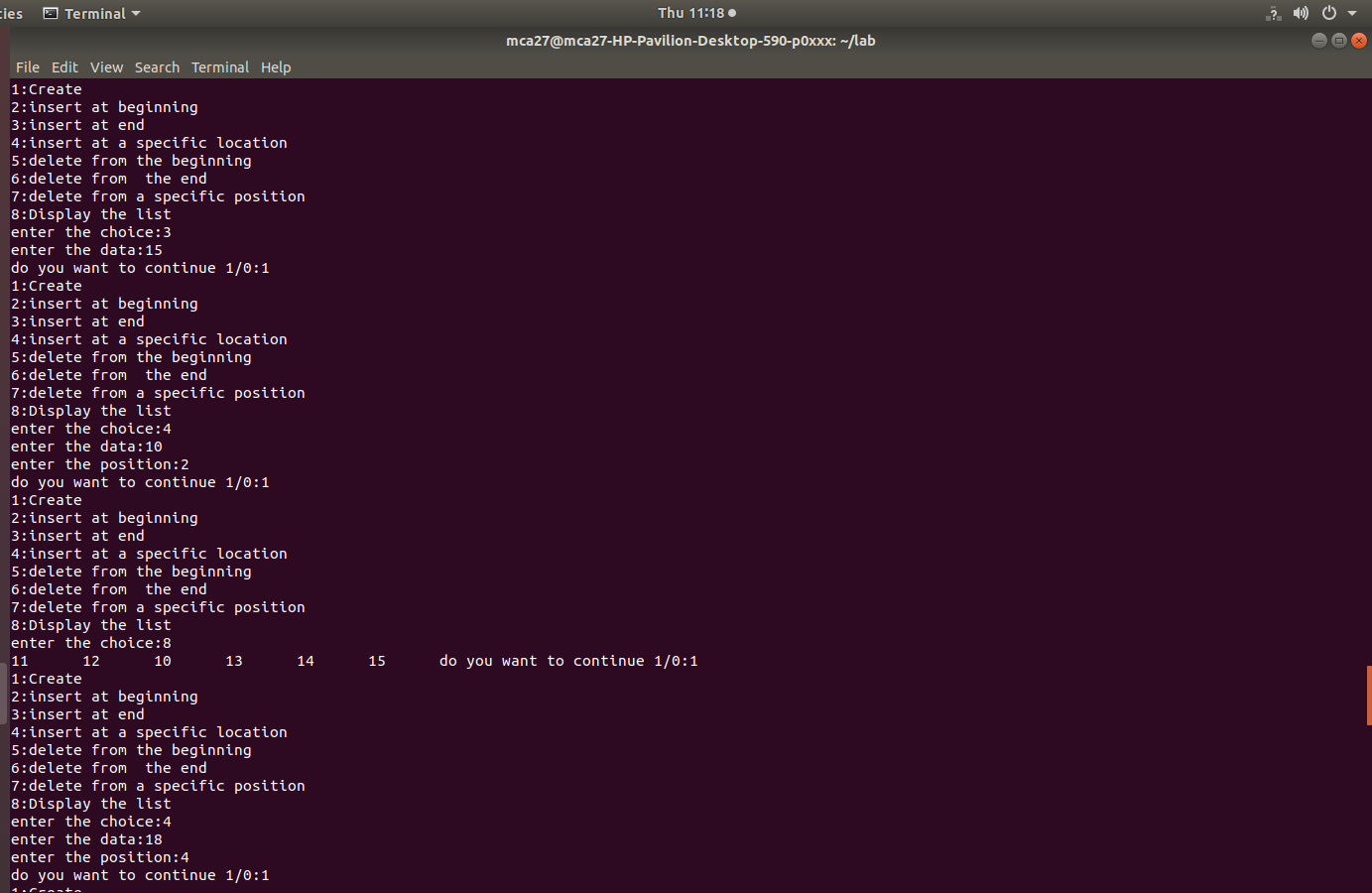
}

}

}

Output:

****

****