Name: Neha Cathrin A

USN: 1BM19CS099

Sec: 3B

Subject: Data Structure Lab

Academic year: 2020

Lab Program 1:

Write a program to simulate the working of stack using an array with the following : a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#define STACK\_SIZE 5

int top=-1;

void push(int item,int s[])

{

if(top==STACK\_SIZE -1)

{

printf("stack overflow \n");

return;

}

top=top+1;

s[top]=item;

}

int pop(int s[])

{

if(top==-1)

{

printf("stack underflow \n");

return(-1);

}

return(s[top--]);

}

void display(int s[])

{

if(top==-1)

{

printf("empty stack \n");

return;

}

printf("contents of stack :\n");

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

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

}

void main()

{

int item,n,s[10],item\_del;

for(;;)

{

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

scanf("%d",&n);

switch(n)

{

case 1:printf("enter item \n");

scanf("%d ",&item);

push(item,s);

break;

case 2:item\_del=pop(s);

if(item\_del==-1)

printf("empty stack \n");

else

printf("deleted item = %d \n",item\_del);

break;

case 3:display(s);

break;

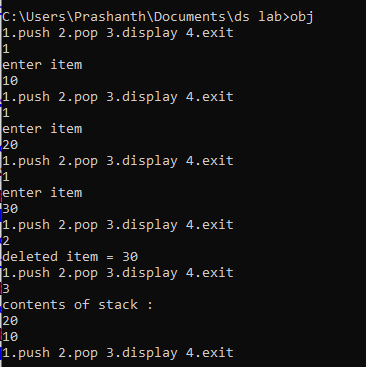
default:exit(0);

}

}

getch();

}



Lab Program 2:

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), \* (multiply) and / (divide) 1

#include<stdio.h>

#include<string.h>

#include<conio.h>

int F(char symbol)

{

switch(symbol)

{

case '+':

case '-':return(2);

case '\*':

case '/':return(4);

case '^':

case '$':return(5);

case '(':return(0);

case '#':return(-1);

default:return(8);

}

}

int G(char symbol)

{

switch(symbol)

{

case '+':

case '-':return(1);

case '\*':

case '/':return(3);

case '^':

case '$':return(6);

case '(':return(9);

case ')':return(0);

default:return(7);

}

}

void infix\_postfix(char infix[],char postfix[])

{

int top,i,j;

char s[30],symbol;

top=-1;

j=0;

s[++top]='#';

for(i=0;i<strlen(infix);i++)

{

symbol=infix[i];

while(F(s[top])>G(symbol))

{

postfix[j]=s[top--];

j++;

}

if(F(s[top])!=G(symbol))

s[++top]=symbol;

else

top--;

}

while(s[top]!='#')

postfix[j++]=s[top--];

postfix[j]='\0';

}

void main()

{

char infix[20];

char postfix[20];

printf("enter infix expression \n");

gets(infix);

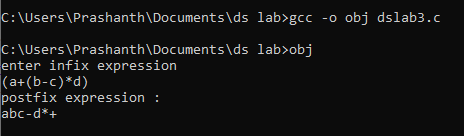
infix\_postfix(infix,postfix);

printf("postfix expression :\n");

puts(postfix);

getch();

}



Lab Program 3:

WAP to simulate the working of a queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<stdlib.h>

#define QUE\_SIZE 5

int item,front=0,rear=-1,q[10];

void insert()

{

if(rear==QUE\_SIZE -1)

{

printf("queue overflow \n");

return;

}

rear=rear+1;

q[rear]=item;

}

int delete()

{

if(front>rear)

{

front=0;

rear=-1;

return(-1);

}

return(q[front++]);

}

void display()

{

if(front>rear)

{

printf("queue is empty \n");

return;

}

printf("contents of queue :\n");

for(int i=front;i<=rear;i++)

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

}

void main()

{

int n;

for(;;)

{

printf("1.insert into queue \n2.delete from queue \n3.display \n4.exit\n");

scanf("%d",&n);

switch(n)

{

case 1:printf("enter item \n");

scanf("%d",&item);

insert();

break;

case 2:item=delete();

if(item==-1)

printf("queue is empty\n");

else

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

break;

case 3:display();

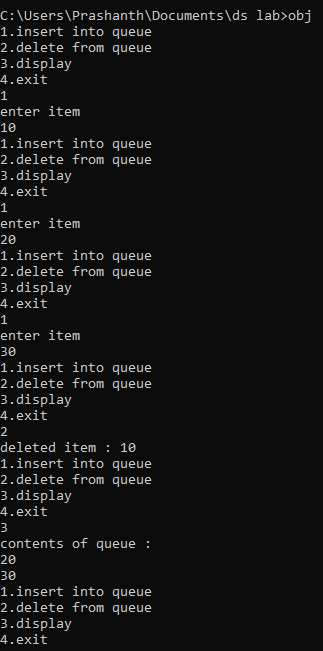
break;

default:exit(0);

}

}

}



Lab program 4:

WAP to simulate the working of a Circular queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

/\*circular queue\*/

#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<stdlib.h>

#define QUE\_SIZE 5

int item,front=0,rear=-1,q[10],count=0;

void insert()

{

if(count==QUE\_SIZE)

{

printf("queue overflow \n");

return;

}

rear=(rear+1)%QUE\_SIZE;

q[rear]=item;

count++;

}

int delete()

{

if(count==0)

{

return(-1);

}

item=q[front];

front=(front+1)%QUE\_SIZE;

count--;

return(item);

}

void display()

{

if(count==0)

{

printf("queue is empty \n");

return;

}

printf("contents of queue :\n");

int f=front;

for(int i=1;i<=count;i++)

{

printf("%d \n",q[f]);

f=(f+1)%QUE\_SIZE;

}

}

void main()

{

int n;

for(;;)

{

printf("1.insert into queue \n2.delete from queue \n3.display \n4.exit\n");

scanf("%d",&n);

switch(n)

{

case 1:printf("enter item \n");

scanf("%d",&item);

insert();

break;

case 2:item=delete();

if(item==-1)

printf("queue is empty\n");

else

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

break;

case 3:display();

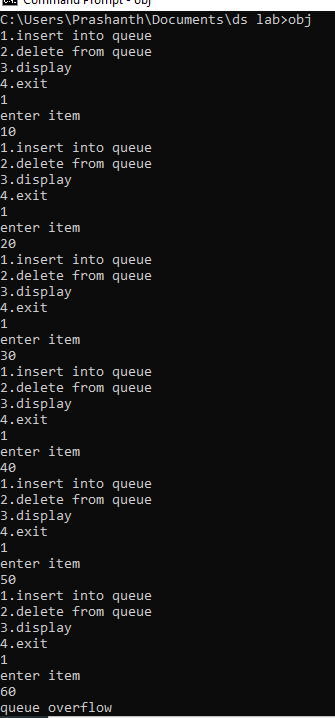
break;

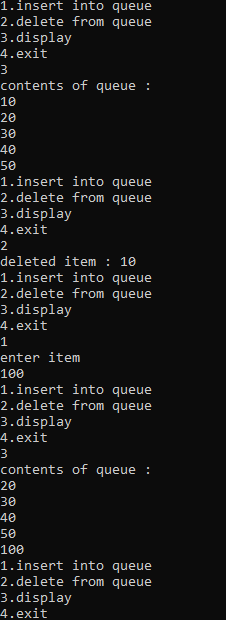
default:exit(0);

}

}

}





Lab Program 5:

WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

#include<process.h>

struct node

{

int info;

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

{

printf("mem full\n");

exit(0);

}

return x;

}

NODE insert\_front(NODE first,int item)

{

NODE temp;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL)

return temp;

temp->link=first;

first=temp;

return first;

}

NODE insert\_rear(NODE first,int item)

{

NODE temp,cur;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL)

return temp;

cur=first;

while(cur->link!=NULL)

cur=cur->link;

cur->link=temp;

return first;

}

NODE insert\_pos(int item,int pos,NODE first)

{

NODE temp;

NODE prev,cur;

int count;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL && pos==1)

return temp;

if(first==NULL)

{

printf("invalid pos\n");

return first;

}

if(pos==1)

{

temp->link=first;

return temp;

}

count=1;

prev=NULL;

cur=first;

while(cur!=NULL && count!=pos)

{

prev=cur;

cur=cur->link;

count++;

}

if(count==pos)

{

prev->link=temp;

temp->link=cur;

return first;

}

printf("IP\n");

return first;

}

void display(NODE first)

{

NODE temp;

if(first==NULL)

printf("list empty cannot display items\n");

for(temp=first;temp!=NULL;temp=temp->link)

{

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

}

}

void main()

{

int item,choice,pos;

NODE first=NULL;

for(;;)

{

printf("\n 1:Insert at front\n 2:Insert at rear\n 3:insert at a position\n 4:display the linked list \n 5:Exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("enter the item at front-end\n");

scanf("%d",&item);

first=insert\_front(first,item);

break;

case 2:printf("enter the item at rear-end\n");

scanf("%d",&item);

first=insert\_rear(first,item);

break;

case 3:printf("enter the position\n");

scanf("%d",&pos);

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

scanf("%d",&item);

first=insert\_pos(item,pos,first);

break;

case 4:display(first);

break;

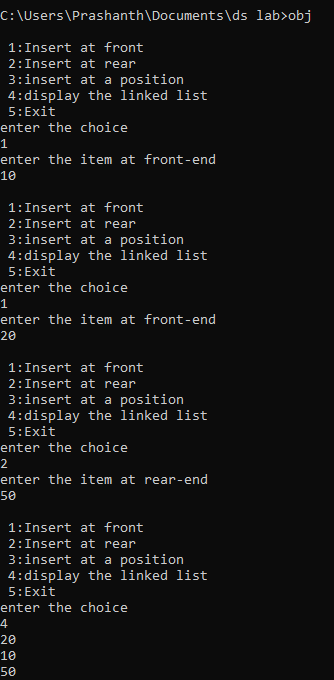
default:exit(0);

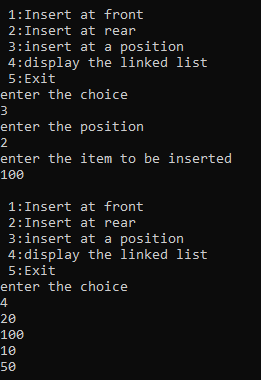
}

}

getch();

}





Lab Program 6:

WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

#include<process.h>

struct node

{

int info;

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

{

printf("mem full\n");

exit(0);

}

return x;

}

NODE insert(NODE first,int item)

{

NODE temp,cur;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL)

return temp;

cur=first;

while(cur->link!=NULL)

cur=cur->link;

cur->link=temp;

return first;

}

NODE delete\_front(NODE first)

{

NODE temp;

if(first==NULL)

{

printf("list is empty \n");

return first;

}

temp=first;

temp=temp->link;

printf("deleted item at front = %d ",first->info);

free(first);

return temp;

}

NODE delete\_rear(NODE first)

{

NODE cur,prev;

if(first==NULL)

{

printf("list is empty \n");

return first;

}

if(first->link==NULL)

{

printf("only one item in list and delete item = %d ",first->info);

free(first);

return NULL;

}

prev=NULL;

cur=first;

while(cur->link!=NULL)

{

prev=cur;

cur=cur->link;

}

printf("deleted item at rear = %d ",cur->info);

free(cur);

prev->link=NULL;

return first;

}

NODE delete\_pos(int pos,NODE first)

{

NODE cur,prev;

int count;

if(first==NULL)

{

printf("list is empty \n");

return first;

}

if(pos<=0)

{

printf("invalid pos value \n");

return first;

}

if(pos==1)

{

cur=first;

first=first->link;

printf("deleted item at position %d = %d ",pos,cur->info);

free(cur);

return first;

}

prev=NULL;

cur=first;

count=1;

while(cur->link!=NULL)

{

if(count==pos)

break;

prev=cur;

cur=cur->link;

count++;

}

if(count!=pos)

{

printf("invalid pos value \n");

return first;

}

prev->link=cur->link;

printf("deleted item at position %d = %d ",pos,cur->info);

free(cur);

return first;

}

void display(NODE first)

{

NODE temp;

if(first==NULL)

printf("list empty cannot display items\n");

for(temp=first;temp!=NULL;temp=temp->link)

{

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

}

}

void main()

{

int item,choice,pos;

NODE first=NULL;

for(;;)

{

printf("\n 1:Insert at rear\n 2:Delete at front\n 3:Delete at rear\n 4:Delete item at a position\n 5:display the linked list \n 6:Exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("enter the item \n");

scanf("%d",&item);

first=insert(first,item);

break;

case 2:

first=delete\_front(first);

break;

case 3: first=delete\_rear(first);

break;

case 4:printf("enter the position\n");

scanf("%d",&pos);

first=delete\_pos(pos,first);

break;

case 5:display(first);

break;

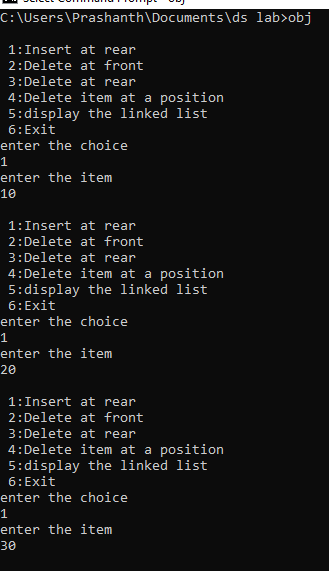
default:exit(0);

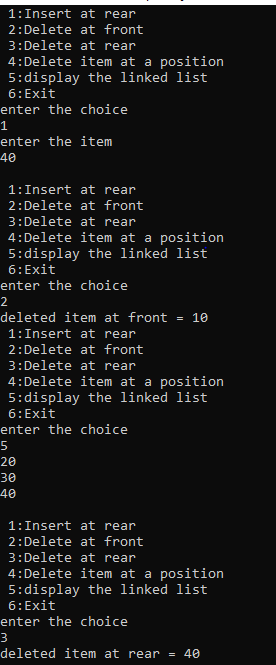
}

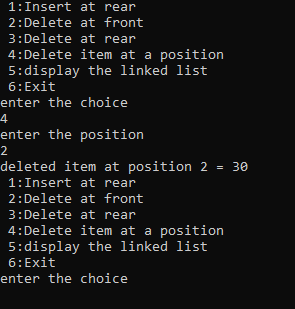
}

getch();

}







Lab Program 7 and Lab Program 8:

WAP Implement Single Link List with following operations a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists d) Stack and Queue Implementation

/\*singly linked list operations 1.sorting 2.reversing 3.concatenating 4.stack queue implementation \*/

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

#include<process.h>

struct node

{

int info;

struct node \*link;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

{

printf("mem full\n");

exit(0);

}

return x;

}

NODE insert\_rear(NODE first,int item)

{

NODE temp,cur;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL)

return temp;

cur=first;

while(cur->link!=NULL)

cur=cur->link;

cur->link=temp;

return first;

}

void display(NODE first)

{

NODE temp;

if(first==NULL)

printf("list is empty");

printf("contents : \n");

for(temp=first;temp!=NULL;temp=temp->link)

{

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

}

}

NODE sort(NODE first)

{

int swapped;

NODE ptr1;

NODE lptr = NULL;

if (first == NULL)

return NULL;

do

{

swapped = 0;

ptr1 = first;

while (ptr1->link != lptr)

{

if (ptr1->info > ptr1->link->info)

{

int tem = ptr1->info;

ptr1->info = ptr1->link->info;

ptr1->link->info = tem;

swapped = 1;

}

ptr1 = ptr1->link;

}

lptr = ptr1;

} while (swapped);

}

NODE reverse(NODE first)

{

NODE cur,temp;

cur=NULL;

while(first!=NULL)

{

temp=first;

first=first->link;

temp->link=cur;

cur=temp;

}

return cur;

}

NODE concat(NODE first,NODE second)

{

NODE cur;

if(first==NULL)

return second;

if(second==NULL)

return first;

cur=first;

while(cur->link!=NULL)

cur=cur->link;

cur->link=second;

return first;

}

NODE delete\_front(NODE first)

{

NODE temp;

if(first==NULL)

{

printf("list is empty \n");

return first;

}

temp=first->link;

printf("deleted item at front = %d\n ",first->info);

free(first);

return temp;

}

NODE delete\_rear(NODE first)

{

NODE cur,prev;

if(first==NULL)

{

printf("list is empty \n");

return first;

}

if(first->link==NULL)

{

printf("only one item in list and delete item = %d ",first->info);

free(first);

return NULL;

}

prev=NULL;

cur=first;

while(cur->link!=NULL)

{

prev=cur;

cur=cur->link;

}

printf("deleted item at rear = %d \n ",cur->info);

free(cur);

prev->link=NULL;

return first;

}

void main()

{

int item,choice,ch,n;

NODE first=NULL,a,b;

NODE stack\_first=NULL,queue\_first=NULL;

for(;;)

{

printf("1.insert\_rear\n2.sorting\n3.display list \n4.concatenating 2 lists \n5.reversing list \n6.stack implementation\n7.queue implementation\n8.exit\n");

printf("enter choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("Enter the item\n");

scanf("%d",&item);

first=insert\_rear(first,item);

break;

case 2:sort(first);

display(first);

break;

case 3:display(first);

break;

case 4:printf("Enter the no of nodes in 1\n");

scanf("%d",&n);

a=NULL;

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

{

printf("Enter the item\n");

scanf("%d",&item);

a=insert\_rear(a,item);

}

printf("Enter the no of nodes in 2\n");

scanf("%d",&n);

b=NULL;

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

{

printf("Enter the item\n");

scanf("%d",&item);

b=insert\_rear(b,item);

}

a=concat(a,b);

display(a);

break;

case 5:first=reverse(first);

display(first);

break;

case 6:printf("Stack\n");

for(;;)

{

printf("\n 1:Insert\_rear\n 2:Delete\_rear\n 3:Display\_list\n 4:Exit\n");

printf("Enter the choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("Enter the item at rear-end\n");

scanf("%d",&item);

first=insert\_rear(first,item);

break;

case 2:first=delete\_rear(first);

break;

case 3:display(first);

break;

default:ch=0;

}

if(ch==0)

break;

}

break;

case 7: printf("QUEUE\n");

for(;;)

{

printf("\n 1:Insert\_rear\n 2:Delete\_front\n 3:Display\_list\n 4:Exit\n");

printf("Enter the choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("Enter the item at rear-end\n");

scanf("%d",&item);

first=insert\_rear(first,item);

break;

case 2:first=delete\_front(first);

break;

case 3:display(first);

break;

default:ch=0;

}

if(ch==0)

break;

}

break;

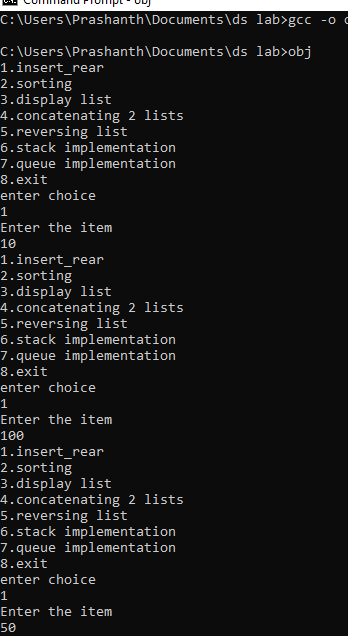
default:exit(0);

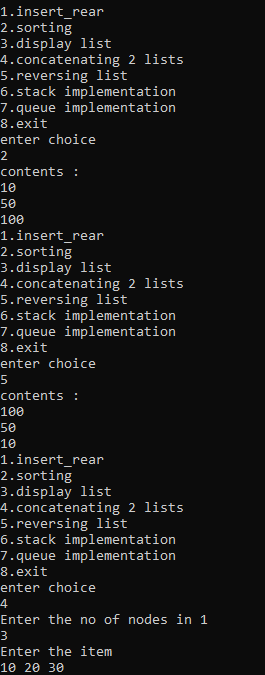
}

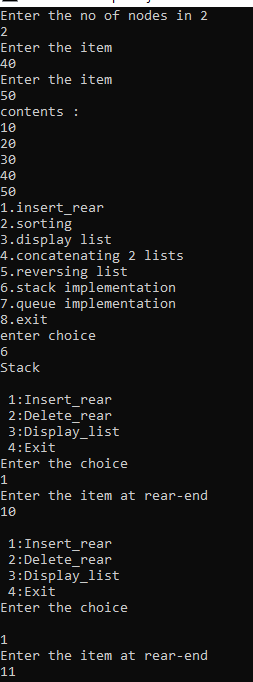
}

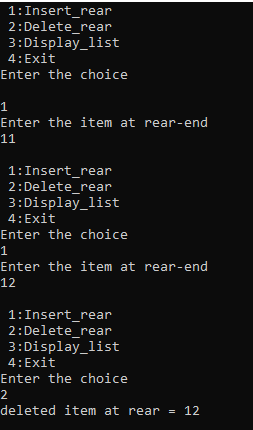
getch();

}









Lab Program 9:

WAP Implement doubly link list with primitive operations a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value d) Display the contents of the list

/\*doubly linked list inserting at end , deleting at a position and display \*/

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

#include<process.h>

struct node

{

int info;

struct node \*llink;

struct node \*rlink;

};

typedef struct node \*NODE;

NODE getnode()

{

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL)

{

printf("mem full\n");

exit(0);

}

return x;

}

NODE insert\_rear(int item,NODE head)

{

NODE temp,cur;

temp=getnode();

temp->info=item;

cur=head->llink;

head->llink=temp;

temp->rlink=head;

temp->llink=cur;

cur->rlink=temp;

return head;

}

NODE insert\_leftpos(int item,NODE head)

{

NODE temp,cur,prev;

if(head->rlink==head)

{

printf("list empty\n");

return head;

}

cur=head->rlink;

while(cur!=head)

{

if(item==cur->info)break;

cur=cur->rlink;

}

if(cur==head)

{

printf("key not found\n");

return head;

}

prev=cur->llink;

printf("enter item towards left of %d=",item);

temp=getnode();

scanf("%d",&temp->info);

prev->rlink=temp;

temp->llink=prev;

cur->llink=temp;

temp->rlink=cur;

return head;

}

NODE delete\_position(int pos,NODE head)

{

NODE p,q;

int c=0;

if(head==NULL)

{

printf("empty list \n");

return head;

}

p=head;

while((p->rlink!=NULL)&&(c!=pos))

{

q=p;

p=p->rlink;

c++;

}

if(c==pos)

{

printf("deleted item at %d = %d ",pos,p->info);

q->rlink=p->rlink;

if(p->rlink!=NULL)

(p->rlink)->llink=q;

free(p);

}

else

printf("invalid position \n");

return head;

}

void display(NODE head)

{

if(head->rlink==head)

{

printf("empty list \n");

}

printf("contents of list : \n");

NODE temp;

temp=head->rlink;

while(temp!=head)

{

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

temp=temp->rlink;

}

}

void main()

{

NODE head;

int item, choice,pos;

head=getnode();

head->rlink=head;

head->llink=head;

for(;;)

{

printf("\n 1:Insert at rear\n 2:insert to left of key item \n 3:Delete at a position\n 4:display the linked list \n 5:Exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("enter the item \n");

scanf("%d",&item);

head=insert\_rear(item,head);

break;

case 2:printf("enter the key item \n");

scanf("%d",&item);

head=insert\_leftpos(item,head);

break;

case 3:printf("enter the position\n");

scanf("%d",&pos);

head=delete\_position(pos,head);

break;

case 4:display(head);

break;

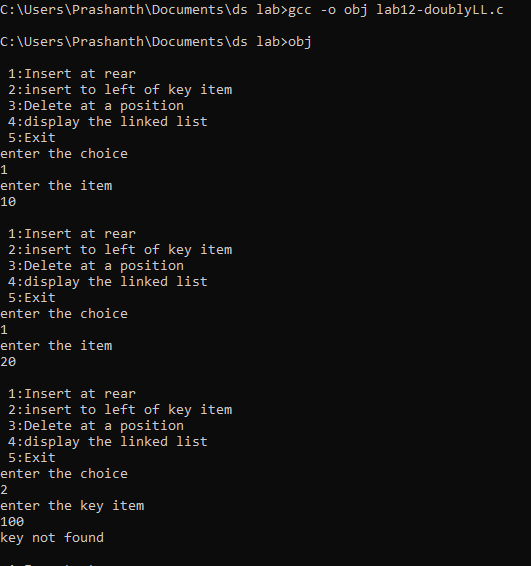
default:exit(0);

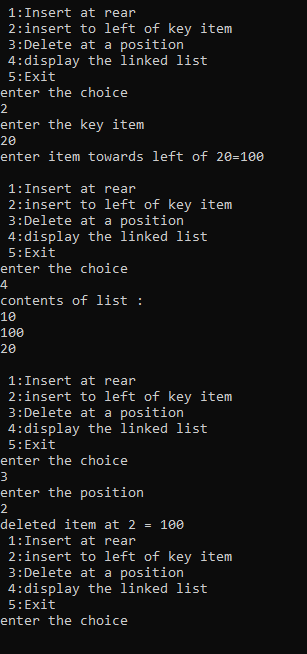
}

}

getch();

}





Lab Program 10:

Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and post order c) To display the elements in the tree.

/\*a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.\*/

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*left;

struct node \*right;

};

typedef struct node \*NODE;

NODE getnode(int item)

{

NODE x = (NODE)malloc(sizeof(struct node));

if(x!=NULL){

x->data=item;

x->left = NULL;

x->right = NULL;

return x;

}

else {

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

exit(0);

}

}

NODE insert(NODE root,int item)

{

if(root ==NULL)

return getnode(item);

if(item<root->data)

root->left = insert(root->left,item);

else if(item>root->data)

root->right = insert(root->right,item);

return root;

}

void inorder(NODE root)

{

if(root == NULL)

return;

inorder(root->left);

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

inorder(root->right);

}

void preorder(NODE root)

{

if(root == NULL)

return;

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

preorder(root->left);

preorder(root->right);

}

void postorder(NODE root)

{

if(root == NULL)

return;

postorder(root->left);

postorder(root->right);

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

}

int main()

{

NODE root = NULL;

int item,ch;

for(;;)

{

printf("1.Insert.\n2.Inorder Traversal.\n3.Preorder Traversal.\n4.Postorder Traversal.\n5.Exit:\n");

scanf("%d",&ch);

switch(ch){

case 1: printf("\nEnter the element:\n");

scanf("%d",&item);

root = insert(root,item);

break;

case 2: inorder(root);

break;

case 3: preorder(root);

break;

case 4: postorder(root);

break;

case 5: exit(1);

default :printf("Invalid Choice");

}

}

}

