1. Create function called swap ( ), which swaps the number values. Create a function pointer which points to a swap ( ) function and call function using pointer. Write a program which also checks whether the two number entered by user is palindrome or not after swaping.

#include<stdio.h>

void swap(int \*a,int \*b)

{

int tmp;

tmp=\*a;

\*a=\*b;

\*b=tmp;

}

int reverse(int n)

{

int rem,rnum=0;

while(n > 0)

{

rem= n % 10;

rnum=rnum \* 10 + rem;

n= n / 10;

}

return rnum;

}

int main()

{

FILE \*fp;

fp=fopen("prog1.txt","r");

int x,y;

printf("\nEnter value of x:");

scanf("%d",&x);

fscanf(fp,"%d",&x);

printf("\nEnter value of y:");

scanf("%d",&y);

fscanf(fp,"%d",&y);

printf("Value before swap \nx= %d \ny= %d",x,y);

void (\*pfun) (int \*,int \*)=swap;

pfun(&x,&y);

printf("\nValue after swap \nx= %d \ny= %d",x,y);

int (\*rp) (int )=reverse;

if(x == rp(x))

printf("\n%d number is palindogram",x);

else

printf("\n%d number is not palindogram",x);

if(y == rp(y))

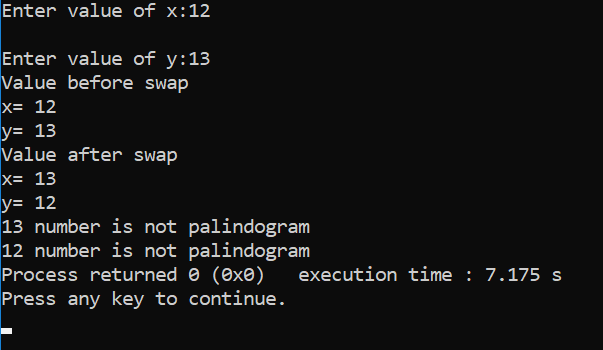
printf("\n%d number is palindogram",y);

else

printf("\n%d number is not palindogram",y);

fclose(fp);

}



3. Write a program to check the balance of parenthesis if an expression. Implement required data structure for the same.

-->#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int top = -1;

char stack[100];

void push(char);

void pop();

void find\_top();

void main()

{

int i;

char a[100];

printf("enter expression\n");

scanf("%s", &a);

for (i = 0; a[i] != '\0';i++)

{

if (a[i] == '(')

{

push(a[i]);

}

else if (a[i] == ')')

{

pop();

}

}

find\_top();

}

void push(char a)

{

stack[top] = a;

top++;

}

void pop()

{

top--;

}

void find\_top()

{

FILE \*ptr;

ptr=fopen("expression.txt","w");

if (top == -1)

{

printf("\nexpression is valid\n",top);

fprintf(ptr,"expression is valid");

}

else

{

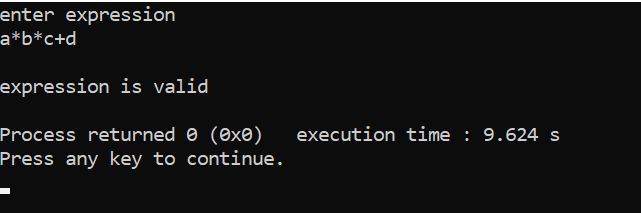
printf("\nexpression is invalid\n");

fprintf(ptr,"expression is invalid");

}

fclose(ptr);

}



4. Implement a program to generate a linked list. For any unsorted linked list, write a method that will delete any duplicates from the linked list without using a temporary buffer.

-->

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \*next;

};

void create(struct node \*\*);

void dup\_delete(struct node \*\*);

void release(struct node \*\*);

void display(struct node \*);

void read(struct node \*);

void write(struct node \*);

int main()

{

struct node \*p = NULL;

struct node\_occur \*head = NULL;

read(&p);

int n;

printf("Enter data into the list\n");

create(&p);

write(&p);

printf("Displaying the nodes in the list:\n");

display(p);

printf("Deleting duplicate elements in the list...\n");

dup\_delete(&p);

printf("Displaying non-deleted nodes in the list:\n");

display(p);

release(&p);

return 0;

}

void dup\_delete(struct node \*\*head)

{

struct node \*p, \*q, \*prev, \*temp;

p = q = prev = \*head;

q = q->next;

while (p != NULL)

{

while (q != NULL && q->num != p->num)

{

prev = q;

q = q->next;

}

if(q == NULL)

{

p = p->next;

if (p != NULL)

{

q = p->next;

}

}

else if (q->num == p->num)

{

prev->next = q->next;

temp = q;

q = q->next;

free(temp);

}

}

}

void create(struct node \*\*head)

{

int c, ch;

struct node \*temp, \*rear;

do

{

printf("Enter number: ");

scanf("%d", &c);

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

temp->num = c;

temp->next = NULL;

if (\*head == NULL)

{

\*head = temp;

}

else

{

rear->next = temp;

}

rear = temp;

printf("Do you wish to continue [1/0]: ");

scanf("%d", &ch);

} while (ch != 0);

printf("\n");

}

void display(struct node \*p)

{

while (p != NULL)

{

printf("%d\t", p->num);

p = p->next;

}

printf("\n");

}

void write(struct node \*p)

{

int data,i;

FILE \*ptr;

ptr=fopen("sortlink.txt","w");

while (p != NULL)

{

fprintf(ptr,"%d\t", p->num);

p = p->next;

}

fclose(ptr);

}

void read(struct node \*p)

{

int i;

FILE \*fptr;

fptr=fopen("sortlink.txt","r");

while (p != NULL)

{

fscanf(fptr,"%d\t", &p->num);

p = p->next;

}

fclose(fptr);

printf("\nRead successufully");

}

void release(struct node \*\*head)

{

struct node \*temp = \*head;

\*head = (\*head)->next;

while ((\*head) != NULL)

{

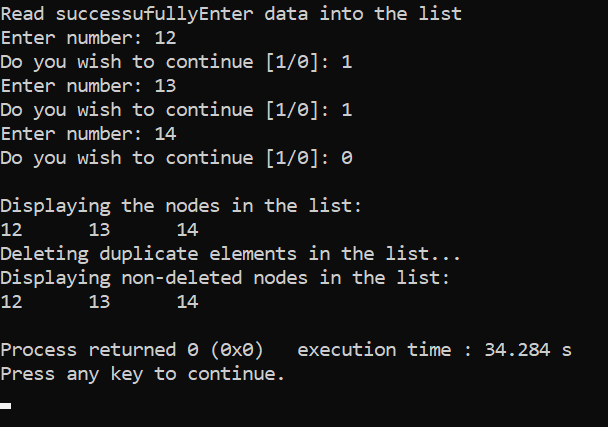
free(temp);

temp = \*head;

(\*head) = (\*head)->next;

}

}



1. Write a program to create a binary tree. Implement required method to generate a binary tree from user inputs and to display binary tree using level order and pre order traversals.

-->#include<stdio.h>

#include<stdlib.h>

struct tree

{

int data;

struct tree \*left;

struct tree \*right;

};

struct tree \*root=NULL;

struct tree \*create\_tree(struct tree \*info,int no)

{

if(info==NULL)

{

info=(struct tree \*)malloc(sizeof(struct tree));

info->data=no;

info->left=NULL;

info->right=NULL;

}

else

{

if(no<=info->data)

{

info->left=create\_tree(info->left,no);

}

else

{

info->right=create\_tree(info->right,no);

}

}

}

void pre\_order(struct tree \*info)

{

if(info!=NULL)

{

printf(" %d ",info->data);

pre\_order(info->left);

pre\_order(info->right);

}

}

void level\_order(struct tree\* info)

{

int h = height(info);

int i;

for (i=1; i<=h; i++)

printGivenLevel(info, i);

}

void printGivenLevel(struct tree\* info, int level)

{

if (info == NULL)

return;

if (level == 1)

printf("%d ", info->data);

else if (level > 1)

{

printGivenLevel(info->left, level-1);

printGivenLevel(info->right, level-1);

}

}

int height(struct tree\* info)

{

if (info==NULL)

return 0;

else

{

int lheight = height(info->left);

int rheight = height(info->right);

if (lheight > rheight)

return(lheight+1);

else

return(rheight+1);

}

}

int main()

{

int ch,n;

do

{

printf("\n1.Create\_tree");

printf("\n2.Preorder");

printf("\n3.Levelorder");

printf("\n0.Exit");

printf("\nEnter Your Choce");

scanf("%d",&ch);

switch(ch)

{

case 1 :printf("\nEnter Value :");

scanf("%d",&n);

root=create\_tree(root,n);

break;

case 2 :pre\_order(root);

break;

case 3 :level\_order(root);

break;

case 0 :exit(0);

break;

default:printf("\nInvalid Choice");

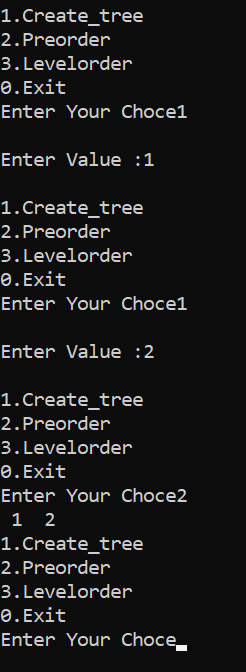
break;

}

}while(ch!=0);

return 0;

}



6.Given two values v1 and v2 (where v1 < v2) within a Binary Search Tree. Print all the keys of tree in range v1 to v2. i.e. print all x such that v1<=x<=v2 and x is a element of given BST. (Create a Binary Search Tree by any method).

--># include<stdio.h>

# include<malloc.h>

struct NODE

{

char Info;

struct NODE \*Left\_Child;

struct NODE \*Right\_Child;

};

int flag = 0;

struct NODE \*Binary\_Tree (char \*, int, int);

int Search\_Node(struct NODE \*, char);

struct NODE \* Binary\_Tree (char \*List, int Lower, int Upper)

{

struct NODE \*Node;

int Mid = (Lower + Upper)/2;

Node = (struct NODE\*) malloc(sizeof(struct NODE));

Node->Info = List [Mid];

if ( Lower>= Upper)

{

Node->Left\_Child = NULL;

Node->Right\_Child = NULL;

return (Node);

}

if (Lower <= Mid - 1)

Node->Left\_Child = Binary\_Tree (List, Lower, Mid - 1);

else

Node->Left\_Child = NULL;

if (Mid + 1 <= Upper)

Node->Right\_Child = Binary\_Tree (List, Mid + 1, Upper);

else

Node->Right\_Child = NULL;

return(Node);

}

int Search\_Node(struct NODE \*Node, char Info)

{

while (Node != NULL)

{

if (Node->Info == Info)

{

flag = 1;

return(flag);

}

else

if(Info < Node->Info)

{

Node = Node->Left\_Child;

}

else

{

Node = Node->Right\_Child;

}

}

return(flag);

}

void main()

{

int flag;

char List[100];

int Number = 0;

char Info ;

char choice;

struct NODE \*T = (struct NODE \*) malloc(sizeof(struct NODE));

T = NULL;

printf("\n Input choice 'b' to break:");

choice = getchar();

while(choice != 'b')

{

fflush(stdin);

printf("\n Input information of the node: ");

scanf("%c", &Info);

List[Number++] = Info;

fflush(stdin);

printf("\n Input choice 'b' to break:");

choice = getchar();

}

Number --;

printf("\n Number of elements in the list is %d", Number+1);

T = Binary\_Tree(List, 0, Number);

// Output(T, 1);

fflush(stdin);

printf("\n Input the information of the node to which want to search: ");

scanf("%c", &Info);

flag = Search\_Node(T, Info);

if (flag)

{

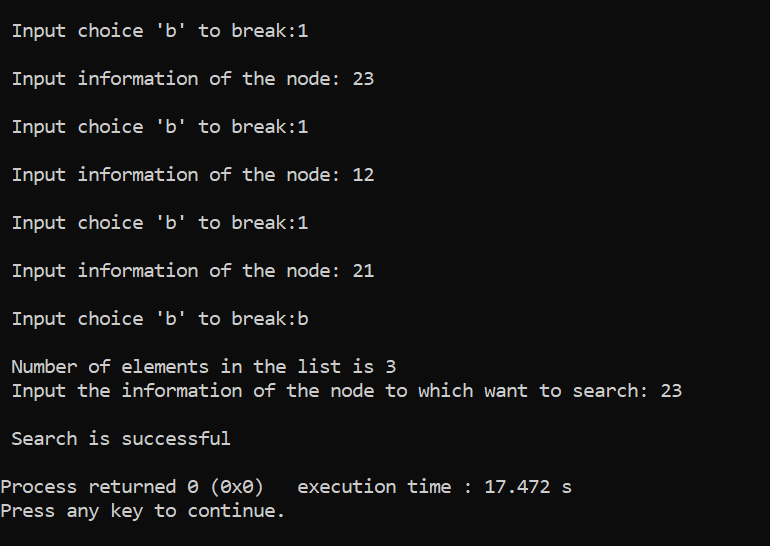
printf("\n Search is successful \n");

}

else

printf("Search unsuccessful");

}



7.Write a program to create a binary tree. Implement required method to generate a binary tree from user inputs and check whether the Binary Tree is a perfect binary tree.

-->#include<stdio.h>

#include<stdlib.h>

struct tree

{

int data;

struct tree \*left;

struct tree \*right;

};

struct tree \*root=NULL;

struct tree \*create\_tree(struct tree \*info,int no)

{

if(info==NULL)

{

info=(struct tree \*)malloc(sizeof(struct tree));

info->data=no;

info->left=NULL;

info->right=NULL;

}

else

{

if(no<=info->data)

{

info->left=create\_tree(info->left,no);

}

else

{

info->right=create\_tree(info->right,no);

}

}

}

void pre\_order(struct tree \*info)

{

if(info!=NULL)

{

printf(" %d ",info->data);

pre\_order(info->left);

pre\_order(info->right);

}

}

void post\_order(struct tree \*info)

{

if(info!=NULL)

{

post\_order(info->left);

post\_order(info->right);

printf(" %d ",info->data);

}

}

void in\_order(struct tree \*info)

{

if(info!=NULL)

{

in\_order(info->left);

printf(" %d ",info->data);

in\_order(info->right);

}

}

int isfulltree(struct tree \*info)

{

if(info==NULL)

return 1;

if(info->left == NULL && info->right == NULL)

return 1;

if((info->left) && (info->right))

return (isfulltree(info->left)&&isfulltree(info->right));

}

int main()

{

int ch,n,l;

do

{

printf("\n1.Create\_tree");

printf("\n2.Inorder");

printf("\n3.Preorder");

printf("\n4.Postorder");

printf("\n5.Prefect Tree or not");

printf("\n6.Exit");

printf("\nEnter Your Choce");

scanf("%d",&ch);

switch(ch)

{

case 1 :printf("\nEnter Value :");

scanf("%d",&n);

root=create\_tree(root,n);

break;

case 2 :in\_order(root);

break;

case 3 :pre\_order(root);

break;

case 4 :post\_order(root);

break;

case 5:if (isfulltree(root))

printf("\nTree is perfect:");

else

printf("\nTree is not perfect");

break;

case 6 :exit(0);

break;

default:printf("\nInvalid Choice");

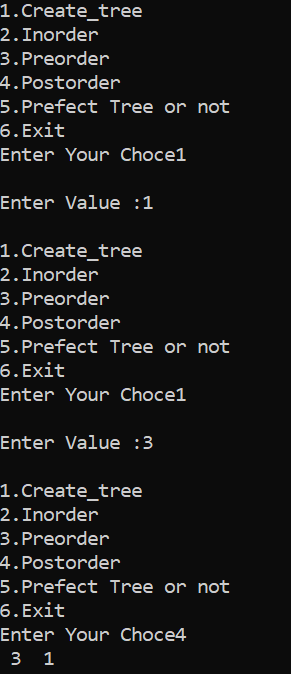
break;

}

}while(ch!=6);

return 0;

}



8. Write a program to implement stack with all basic operations using linked list.

-->#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*top=NULL;

void push(int);

void pop();

void disp();

void read();

void write();

int main()

{

read();

int ch,value;

do

{

printf("\n1.push...");

printf("\n2.pop...");

printf("\n3.display...");

printf("\n4.Exit...");

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter value:");

scanf("%d",&value);

push(value);

write();

break;

case 2: pop();

write();

break;

case 3: disp();

break;

case 4: exit(1);

break;

default:printf("\nWrong Choice...");

break;

}

}while(ch!=4);

}

void push(int x)

{

struct node \*new\_node;

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

new\_node->data=x;

if(top==NULL)

{

new\_node->next=NULL;

}

else

{

new\_node->next=top;

//top=new\_node;

}

top=new\_node;

printf("\nSuccessfully inserted");

}

void pop()

{

if(top==NULL)

printf("\nStacklist is not created");

else

{

struct node \*temp;

temp=top;

printf("\nDeleted emelemnt is %d",temp->data);

top=temp->next;

free(temp);

}

}

void disp()

{

if(top==NULL)

printf("\nStack is empty");

else

{

struct node \*temp;

temp=top;

while(temp!=NULL)

{

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

temp=temp->next;

}

//printf("\nData is %d",temp->data);

}

}

void write()

{

int data,i;

FILE \*ptr;

ptr=fopen("stacklink.txt","w");

struct node \*temp;

temp=top;

while(temp!=NULL)

{

fprintf(ptr,"%d",temp->data);

temp=temp->next;

}

fclose(ptr);

}

void read()

{

int i;

FILE \*fptr;

fptr=fopen("stacklink.txt","r");

struct node \*temp;

temp=top;

while(temp!=NULL)

{

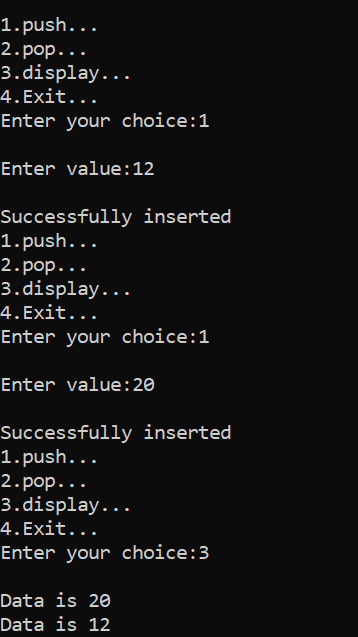
fscanf(fptr,"%d",&temp->data);

temp=temp->next;

}

fclose(fptr);

}



9.Write a program to implement Queue with all basic operations using linked list.

-->#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*front=NULL,\*rear=NULL;

void equeue(int);

void dqueue();

void disp();

void read();

void write();

int main()

{

read();

int ch,value;

printf("\n1.Equeue");

printf("\n2.Dqueue");

printf("\n3.Display");

printf("\n4.Exit");

do

{

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("\nEnter value:");

scanf("%d",&value);

equeue(value);

write();

break;

case 2:dqueue();

write();

break;

case 3:disp();

break;

default:

break;

}

}while(ch!=4);

}

void equeue(int x)

{

struct node \*new\_node;

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

new\_node->data=x;

new\_node->next=NULL;

if(rear==NULL)

front=rear=new\_node;

else

{

rear->next=new\_node;

rear=new\_node;

}

printf("\nSuccessfull inserted");

}

void dqueue()

{

if(front==NULL)

printf("\nQueue is empty");

else

{

struct node \*temp;

temp=front;

printf("\nDeleted element is %d",temp->data);

front=temp->next;

free(temp);

}

}

void disp()

{

if(front==NULL)

printf("\nQueue is empty");

else

{

struct node \*temp;

temp=front;

while(temp!=NULL)

{

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

temp=temp->next;

}

}

}

void write()

{

int data,i;

FILE \*ptr;

ptr=fopen("queuelink.txt","w");

struct node \*temp;

temp=front;

while(temp!=NULL)

{

fprintf(ptr,"%d",temp->data);

temp=temp->next;

}

fclose(ptr);

}

void read()

{

int i;

FILE \*fptr;

fptr=fopen("queuelink.txt","r");

struct node \*temp;

temp=front;

while(temp!=NULL)

{

fscanf(fptr,"%d",&temp->data);

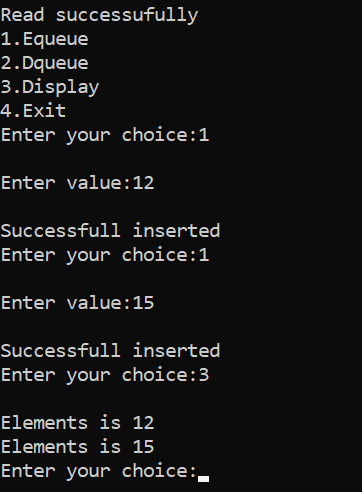
temp=temp->next;

}

fclose(fptr);

printf("\nRead successufully");

}



1. Write a program to implement stack with required operations using array.

-->#include<stdio.h>

#include<conio.h>

#define size 5

int stack[size];

int top=-1;

void push();

void display();

void isEmpty();

void pop();

void update();

int main()

{

read();

int ch;

printf("\n1) push operation");

printf("\n2) pop operation");

printf("\n3) peep operation");

printf("\n4) update operation");

printf("\n5) isEmpty operation");

printf("\n6) exit");

do

{

printf("\n enter your choice=");

scanf("%d",&ch);

switch(ch)

{

case 1:

push();

write();

break;

case 2:

pop();

write();

break;

case 3:

peep();

break;

case 4:

update();

write();

break;

case 5:

isEmpty();

break;

default:

return 0;

break;

}

}while(ch!=5);

}

void write()

{

int data,i;

FILE \*ptr;

ptr=fopen("stack.txt","w");

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

fprintf(ptr,"%d\t",stack[i]);

fclose(ptr);

}

void read()

{

int i;

FILE \*fptr;

fptr=fopen("stack.txt","r");

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

fscanf(fptr,"%d\t",&stack[i]);

fclose(fptr);

}

void push()

{

int data;

if(top>=size-1)

{

printf("\n stack is overflow");

}

else

{

printf("\Enter an element=");

scanf("%d",&data);

top++;

stack[top]=data;

}

}

void peep()

{

int i;

if(top==-1)

{

printf("\n stack is underflow");

}

else

{

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

{

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

}

}

}

void pop()

{

if(top==-1)

{

printf("\n stack is underflow");

}

else

{

printf("\n popped element is=%d",stack[top]);

stack[top--];

}

}

void update()

{

int u,n;

if(top==-1)

{

printf("\n stack is underflow");

}

else

{

printf("\n enter position at you want to change=");

scanf("%d",&u);

printf("\n enter new element= ");

scanf("%d",&n);

stack[u]=n;

}

}

void isEmpty()

{

if(top==-1)

{

printf("\n stack is empty");

}

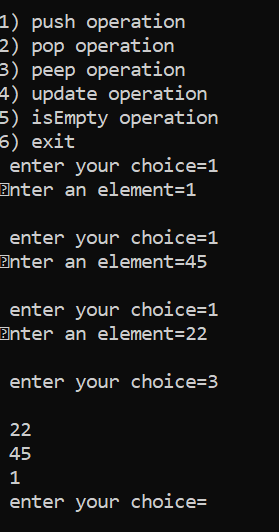
else

{

printf("\n stack is not empty");

}

}



1. Write a program to implement Queue with required operations using array.

-->#include<stdio.h>

#include<conio.h>

#include<ctype.h>

#include<string.h>

#include<stdlib.h>

#define size 10

int ch;

int q[size];

int rear=-1;

int front=-1;

void insert\_queue();

void delete\_queue();

void disp\_queue();

void insert\_queue()

{

if(rear>=size)

printf("\nQueue is overflow");

else

{

if(front==-1)

front=0;

printf("\nEnter element:");

scanf("%d",&ch);

rear=rear+1;

q[rear]=ch;

printf("\nInserted successfully");

}

}

void delete\_queue()

{

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

printf("\nQueue underflow");

else

{

ch=q[front];

printf("\nDelete Element is %d",ch);

front=front+1;

}

}

void disp\_queue()

{

int i;

if(front==-1)

return;

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

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

}

void write()

{

int data,i;

FILE \*ptr;

ptr=fopen("queue.txt","w");

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

fprintf(ptr,"%d\t",q[i]);

fclose(ptr);

}

void read()

{

int i;

FILE \*fptr;

fptr=fopen("queue.txt","r");

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

fscanf(fptr,"%d\t",&q[i]);

fclose(fptr);

printf("\nRead successufully");

}

void main()

{

read();

int ch;

do

{

printf("\n1.Insert");

printf("\n2.Delete");

printf("\n3.Display");

printf("\n4.Exit");

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:insert\_queue();

write();

break;

case 2:delete\_queue();

write();

break;

case 3:disp\_queue();

break;

case 4:exit(1);

break;

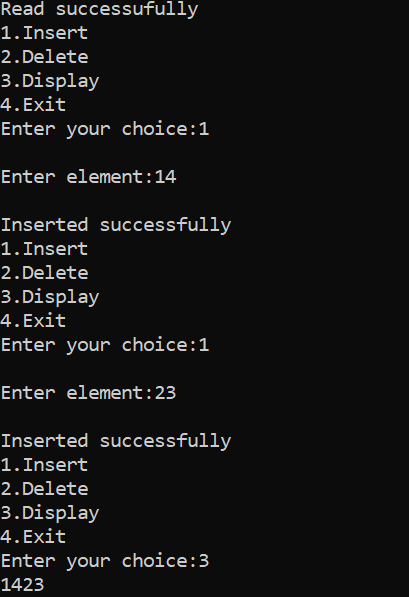
default:

break;

}

}while(ch!=5);

}



12. Write a program to check whether the string is palindrome or not. Use Stack Data Structure for the same.

-->#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 50

int top = -1, front = 0;

int stack[MAX];

void push(char);

void pop();

void main()

{

// read();

FILE \*ptr;

int i, ch;

char s[MAX], b;

do

{

printf("\n1-Enter string....");

printf("\n2-Exit....");

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

scanf("%d",&ch);

switch (ch)

{

case 1:

ptr=fopen("palindrome.txt","w");

printf("Enter the String\n");

scanf("%s", s);

for (i = 0;s[i] != '\0';i++)

{

b = s[i];

push(b);

}

for (i = 0;i < (strlen(s) / 2);i++)

{

if (stack[top] == stack[front])

{

pop();

front++;

}

else

{

printf("%s is not a palindrome\n", s);

fprintf(ptr,"%s is not palindrome",s);

break;

}

}

if ((strlen(s) / 2) == front)

{

printf("%s is palindrome\n", s);

fprintf(ptr,"%s is palindrome",s);

}

front = 0;

top = -1;

fclose(ptr);

//write();

break;

case 2:

exit(0);

default:

printf("enter correct choice\n");

}

}while(ch!=3);

}

void push(char a)

{

top++;

stack[top] = a;

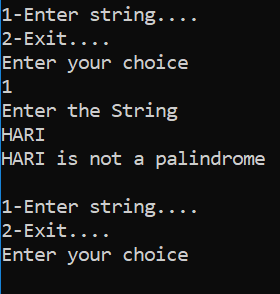
}

void pop()

{

top--;

}



13. Write a program to implement Doubly Linked List.

-->#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

struct node \*prev;

};

struct node \*head=NULL;

void insert\_first()

{

int n;

struct node \*new\_node;

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

printf("\nEnter number:");

scanf("%d",&n);

new\_node->data=n;

new\_node->next=NULL;

new\_node->prev=NULL;

if(head!=NULL)

{

new\_node->next=head;

head->prev=new\_node;

head=new\_node;

}

else

head=new\_node;

}

void insert\_last()

{

int n;

struct node \*new\_node;

struct node \*temp;

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

printf("\nEnter number:");

scanf("%d",&n);

new\_node->data=n;

new\_node->next=NULL;

new\_node->prev=NULL;

temp=head;

if(head==NULL)

printf("\nList is emtpty");

else

{

while(temp->next!=NULL)

temp=temp->next;

temp->next=new\_node;

new\_node->prev=temp;

}

}

void insert\_specific()

{

int n;

struct node \*new\_node;

struct node \*temp;

int p,cnt=1;

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

printf("\nEnter number:");

scanf("%d",&n);

new\_node->data=n;

new\_node->next=NULL;

new\_node->prev=NULL;

printf("\nEnter position:");

scanf("%d",&p);

temp=head;

if(head==NULL)

{

printf("\nList is empty");

}

else

{

while(cnt!=p)

{

temp=temp->next;

cnt++;

}

new\_node->next=temp->next;

temp->next=new\_node;

new\_node->prev=temp;

}

}

void insert()

{

int ch,inch;

printf("\n1.Insert First....");

printf("\n2.Insert Last.....");

printf("\n3.Insert Specific..");

printf("\nEnter your choice...");

scanf("%d",&ch);

if(ch==1)

insert\_first();

else if(ch==2)

insert\_last();

else

insert\_specific();

}

void delete\_first()

{

struct node \*temp;

if(head==NULL)

printf("\nList is empty");

else

{

temp=head;

head=head->next;

head->next->prev=head;

printf("\nDeleted element is %d",temp->data);

free(temp);

}

}

void delete\_last()

{

struct node \*temp;

struct node \*prev;

temp=head;

if(head==NULL)

printf("\nList is empty");

else

{

while(temp->next!=NULL)

{

prev=temp;

temp=temp->next;

}

printf("\nDeleted element is %d",temp->data);

prev->next=NULL;

free(temp);

}

}

void delete()

{

int ch,inch;

printf("\n1.Delete First....");

printf("\n2.Delete Last.....");

printf("\n3.Delete Specific..");

printf("\nEnter your choice...");

scanf("%d",&ch);

if(ch==1)

delete\_first();

else if(ch==2)

delete\_last();

//else

// delete\_specific();

}

void disp()

{

struct node \*new\_node;

struct node \*temp;

temp=head;

while(temp)

{

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

temp=temp->next;

}

}

void write()

{

int data,i;

FILE \*ptr;

ptr=fopen("doublylink.txt","w");

struct node \*temp;

temp=head;

while(temp)

{

fprintf(ptr,"%d",temp->data);

temp=temp->next;

}

fclose(ptr);

}

void read()

{

int i;

FILE \*fptr;

fptr=fopen("doublylink.txt","r");

struct node \*temp;

temp=head;

while(temp)

{

fscanf(fptr,"%d",&temp->data);

temp=temp->next;

}

fclose(fptr);

printf("\nRead successufully");

}

int main()

{

read();

int ch,n;

do

{

printf("\n1.Insert..");

printf("\n2.Delete..");

printf("\n3:Display");

printf("\n4:Exit");

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:insert();

write();

break;

case 2:delete();

write();

break;

case 3:disp();

break;

case 4:exit(1);

break;

default:

break;

}

}while(ch!=4);

return 0;

}

