1.

#include<stdio.h>

#include<stdlib.h>

int a[10],i,n,pos,elem;

void Create()

{

printf("\n enter the size of the array element");

scanf("%d",&n);

printf("\n enter the elements in the array\n");

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

scanf("%d",&a[i]);

}

void Display()

{

int i;

printf("\n the array elements are:\n");

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

{

printf("%d\t",a[i]);

}

}

void Insert()

{

printf("\n enter the position for the new element");

scanf("%d",&pos);

printf("enter the element to be inserted");

scanf("%d",&elem);

for(i=n-1;i>=pos;i--)

{

a[i+1]=a[i];

}

a[pos]=elem;

n=n+1;

}

void Delete()

{

printf("\n enter the position for the element to be deleted");

scanf("%d",&pos);

elem=a[pos];

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

{

a[i]=a[i+1];

}

n=n-1;

printf("\n the deleted element is %d",elem);

}

void main()

{

int ch;

do

{

printf("\n\n --------MENU--------\n\n");

printf("1.Create\n 2.Display\n 3.Insert\n 4.Delete\n 5.exit\n");

printf("----------------");

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

scanf("%d",&ch);

switch(ch)

{

case 1:Create();

break;

case 2:Display();

break;

case 3:Insert();

break;

case 4:Delete();

break;

case 5:exit;

break;

default:printf("\n invalid choice\n");

break;

}

}while(ch!=5);

}

2.

#include<stdio.h>

#include<stdlib.h>

char str[100],pat[50],rep[50],ans[100];

int i,j,k,c,m,flag=0;

void stringmatch()

{

i=j=c=m=0;

while(str[c]!='\0')

{

if(str[m]==pat[i])

{

i++;

m++;

if(pat[i]=='\0')

{

flag=1;

for(k=0;rep[k]!='\0';k++,j++)

ans[j]=rep[k];

i=0;

c=m;

}

}

else

{

ans[j]=str[c];

j++;

c++;

m=c;

i=0;

}

}

ans[j]='\0';

}

void main()

{

printf("\n enter the main string\n");

gets(str);

printf("\n enter the pattern string\n");

gets(pat);

printf("\n enter the replace string\n");

gets(rep);

stringmatch();

if(flag==1)

printf("\n the resultant string is \n%s",ans);

else

printf("\n the string not found\n");

}

3.

#include<stdio.h>

#include<stdlib.h>

int i,top,ch,item,s[3];

void push(),pop(),dis();

void main()

{

top=-1;

ch=0;

while(ch!=4)

{

printf("\n enter the menu for the stack opertion:\n");

printf("\n 1.insert\n 2.delete\n 3.display\n 4.exit\n");

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:push();

break;

case 2:pop();

break;

case 3:dis();

break;

case 4:exit(0);

break;

default:printf("invalid choice\n");

}

}

return;

}

void push()

{

int item;

if(top==3-1)

{

printf("stack is overflow\n");

return;

}

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

scanf("%d",&item);

top+=1;

s[top]=item;

}

void pop()

{

if(top==-1)

{

printf("stack is underflow\n");

return;

}

printf("item popped is %d\n",s[top--]);

}

void dis()

{

if(top==-1)

{

printf("stack is empty\n");

return;

}

printf("\n stack contains......\n");

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

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

}

4.

#include<stdio.h>

#include<string.h>

void push(char item[],int\*top,char s[20][20])

{

\*top+=1;

strcpy(s[\*top],item);

}

char\*pop(int\*top,char s[20][20])

{

char\*item;

item=s[\*top];

\*top-=1;

return item;

}

void pos\_to\_in(char postfix[],char infix[])

{

char s[20][20],symbol,temp[2],\*op1,\*op2;

int i,top;

top=-1;

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

{

symbol=postfix[i];

temp[0]=symbol;

temp[1]='\0';

switch(symbol)

{

case '+':

case '-':

case '\*':

case '/':

case '$':

op2=pop(&top,s);

op1=pop(&top,s);

strcpy(infix,"(");

strcat(infix,op1);

strcat(infix,temp);

strcat(infix,op2);

strcat(infix,")");

push(infix,&top,s);

break;

default:push(temp,&top,s);

}

}

}

int main()

{

char postfix[25],infix[25];

printf("enter your postfix expr\n");

scanf("%s",postfix);

pos\_to\_in(postfix,infix);

printf("\n the equivalent postfix is %s\n",infix);

return 0;

}

5a.

#include<stdio.h>

#include<ctype.h>

#include<math.h>

int stack[25],top=-1;

int push();

int pop();

int eval\_postfix();

int main()

{

char postfix[25];

int i=0,value[20],result;

printf("enter a valid postfix expr\n");

scanf("%s",postfix);

while(postfix[i]!='\0')

{

if(isalpha(postfix[i]))

{

printf("enter the value %c\n",postfix[i]);

scanf("%d",&value[i]);

}

i++;

}

result= eval\_postfix(postfix,value);

printf("the result of %s=%d\n",postfix,result);

}

int eval\_postfix(char postfix[],int data[])

{

int i=0,op1,op2,res,p;

char ch;

while(postfix[i]!='\0')

{

ch=postfix[i];

if(isalpha(ch))

push(data[i]);

else

{

op2=pop();

op1=pop();

switch(ch)

{

case'+':push(op1+op2);

break;

case'-':push(op1-op2);

break;

case'/':push(op1/op2);

break;

case'\*':push(op1\*op2);

break;

case'$':p=pow(op1,op2);

push(p);

break;

}

}

i++;

}

res=pop();

return res;

}

int push(int num)

{

top+=1;

stack[top]=num;

return 0;

}

int pop()

{

int num;

num=stack[top--];

return num;

}

5b.

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

void tower(int n,int source,int temp,int destination)

{

if(n==0)

return;

tower(n-1,source,destination,temp);

printf("\n move disc %d from %c to %c",n,source,destination);

tower(n-1,temp,source,destination);

}

void main()

{

int n;

printf("\nEnter the number of discs:\n");

scanf("%d",&n);

tower(n,'A','B','C');

printf("\n\nTotal number of moves are:%d\n",(int)pow(2,n)-1);

}

6.

#include<stdio.h>

#include<stdlib.h>

int i,front,rear,ch,s[3],item;

void insert(),delete(),dis();

int main()

{

front=0;

rear=-1;

ch=0;

while(ch!=4)

{

printf("\n enter the menu for the queue operation\n");

printf("\n 1.insert\n 2.delete\n 3.display\n 4.exit\n");

printf("input your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:insert();

break;

case 2:delete();

break;

case 3:dis();

break;

case 4:exit(0);

break;

default:printf("invalid choice\n");

}

}

return 0;

}

void insert()

{

int item;

if(rear==3-1)

{

printf("queue is overflow\n");

return;

}

printf("enter an item to inserted:");

scanf("%d",&item);

rear+=1;

s[rear]=item;

}

void delete()

{

if(front>rear)

{

printf("queue is empty\n");

return;

}

printf("item to be deleted is %d\n",s[front++]);

}

void dis()

{

if(front>rear)

{

printf("queue is empty\n");

return;

}

printf("\n queue contains......\n");

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

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

}

7.

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

void create();

void insert();

void delete();

void display();

struct node

{

int data;

struct node \*link;

};

struct node \*first=NULL,\*last=NULL,\*next,\*prev,\*cur;

void create()

{

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

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

scanf("%d",&cur->data);

cur->link=NULL;

first=cur;

last=cur;

}

void insert()

{

int pos,c=1;

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

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

scanf("%d",&cur->data);

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

scanf("%d",&pos);

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

{

cur->link=first;

first=cur;

}

else

{

next=first;

while(c<pos)

{

prev=next;

next=prev->link;

c++;

}

if(prev==NULL)

{

printf("\n invalid position\n");

}

else

{

cur->link=prev->link;

prev->link=cur;

}

}

}

void delete()

{

int pos,c=1;

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

scanf("%d",&pos);

if(first==NULL)

{

printf("\n list is empty\n");

}

else if(pos==1 && first->link==NULL)

{

printf("\n deleted element is %d\n",first->data);

free(first);

first=NULL;

}

else if(pos==1 && first->link!=NULL)

{

cur=first;

first=first->link;

cur->link=NULL;

printf("\n deleted element is %d\n",cur->data);

free(cur);

}

else

{

next=first;

while(c<pos)

{

cur=next;

next=next->link;

c++;

}

cur->link=next->link;

next->link=NULL;

if(next==NULL)

{

printf("\n invalid position\n");

}

else

{

printf("\n deleted element is %d\n",next->data);

free(next);

}

}

}

void display()

{

cur=first;

while(cur!=NULL)

{

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

cur=cur->link;

}

}

void main()

{

int ch;

printf("\n\n singly linked list");

do

{

printf("\n\n 1.create\n 2.insert\n 3.delete\n 4.exit\n");

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

scanf("%d",&ch);

switch(ch)

{

case 1:

create();

display();

break;

case 2:

insert();

display();

break;

case 3:

delete();

display();

break;

case 4:

exit(0);

default:

printf("invalid choice....");

}

}while(1);

}

8.

#include<stdio.h>

#include<malloc.h>

struct node

{

int value;

struct node \*next;

};

int main()

{

int item,n;

struct node \*head;

struct node \*tail;

struct node \*temp;

struct node \*prev;

struct node \*current;

struct node \*next;

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

head=temp;

printf("enter the size of list\n");

scanf("%d",&n);

printf("enter the list to be reversed\n");

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

scanf("%d",&item);

temp->value=item;

head=temp;

n--;

while(n!=0)

{

temp->next=(struct node\*)malloc(sizeof(struct node));

temp=temp->next;

scanf("%d",&item);

temp->value=item;

--n;

}

temp->next=NULL;

tail=temp;

temp=head;

while(temp)

{

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

temp=temp->next;

}

printf("reversing the linked list\n");

prev=NULL;

current=next=head;

while(current)

{

next=current->next;

current->next=prev;

prev=current;

current=next;

}

temp=tail;

while(temp)

{

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

temp=temp->next;

}

return 0;

}

9.

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

typedef struct node

{

int priority;

int info;

struct node \*link;

}node;

node \*front=NULL;

void insert(int item,int priority)

{

node \*temp,\*q;

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

temp->info=item;

temp->priority=priority;

if(front==NULL||priority<front->priority)

{

temp->link=front;

front=temp;

}

else

{

q=front;

while(q->link!=NULL&&q->link->priority<=priority)

q=q->link;

temp->link=q->link;

q->link=temp;

}

}

void del()

{

node \*temp;

if(front==NULL)

printf("queue underflow\n");

else

{

temp=front;

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

front=front->link;

free(temp);

}

}

void display()

{

node \*ptr;

ptr=front;

if(front==NULL)

printf("queue is empty\n");

else

{

printf("queue is:\n");

printf("priority item\n");

while(ptr!=NULL)

{

printf("%5d %5d\n",ptr->priority,ptr->info);

ptr=ptr->link;

}

}

}

int main()

{

int choice,item,priority;

do

{

printf("1.insert\n");

printf("2.delete\n");

printf("3.display\n");

printf("4.quit\n");

printf("enter your choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("input the item value to be added in the queue:");

scanf("%d",&item);

printf("enter its priority:");

scanf("%d",&priority);

insert(item,priority);

break;

case 2:

del();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("wrong choice\n");

}

}

while(choice!=4);

return 0;

}

10.

#include<stdio.h>

#include<stdlib.h>

struct node

{

int value;

struct node \*left;

struct node \*right;

};

struct node \*root;

struct node\* insert(struct node\* r,int data);

void inorder(struct node\* r);

void preorder(struct node\* r);

void postorder(struct node\* r);

int main()

{

root=NULL;

int n,v;

printf("how many data's do you want to insert?\n");

scanf("%d",&n);

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

{

printf("data %d:",i+1);

scanf("%d",&v);

root=insert(root,v);

}

printf("inorder traversal:");

inorder(root);

printf("\n");

printf("preorder traversal:");

preorder(root);

printf("\n");

printf("postorder traversal:");

postorder(root);

printf("\n");

return 0;

}

struct node\* insert(struct node\* r,int data)

{

if(r==NULL)

{

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

r->value=data;

r->left=NULL;

r->right=NULL;

}

else if(data<r->value)

{

r->left=insert(r->left,data);

}

else

{

r->right=insert(r->right,data);

}

return r;

}

void inorder(struct node\* r)

{

if(r!=NULL)

{

inorder(r->left);

printf("%d",r->value);

inorder(r->right);

}

}

void preorder(struct node\* r)

{

if(r!=NULL)

{

printf("%d",r->value);

preorder(r->left);

preorder(r->right);

}

}

void postorder(struct node\* r)

{

if(r!=NULL)

{

postorder(r->left);

postorder(r->right);

printf("%d",r->value);

}

}

11.

#include<stdio.h>

#include<stdlib.h>

#define MAX 10

struct employee

{

int id;

char name[15];

};

typedef struct employee EMP;

EMP emp[MAX];

int a[MAX];

int create(int num)

{

int key;

key=num % 100;

return key;

}

int getemp(EMP emp[],int key)

{

printf("\n enter the emp id:");

scanf("%d",&emp[key].id);

printf("\n enter emp name:");

scanf("%s",&emp[key].name);

return key;

}

void display()

{

int i,ch;

printf("\n 1.Display All\n 2.Filtered Display");

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

scanf("%d",&ch);

if(ch==1)

{

printf("\n The hash table is:\n");

printf("\n HTKey\tEmpID\tEmpName");

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

printf("\n %d\t%d\t%s",i,emp[i].id,emp[i].name);

}

else

{

printf("\n The hash table is:\n");

printf("\n HTKey\tEmpID\tEmpName");

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

if(a[i]!=-1)

{

printf("\n %d\t%d\t%s",i,emp[i].id,emp[i].name);

continue;

}

}

}

void linear\_prob(int key)

{

int flag,i,count=0;

flag=0;

if(a[key]==-1)

{

a[key]=getemp(emp,key);

}

else

{

printf("\n collision Detected....!!!\n");

i=0;

while(i<MAX)

{

if(a[i]!=-1)

count++;

else

i++;

}

printf("\n Collision avoided successfully using LINEAR PROBING\n");

if(count==MAX)

{

printf("\n Hash table is full");

display(emp);

exit(1);

}

for(i=key;i<MAX;i++)

if(a[i]==-1)

{

a[i]=getemp(emp,key);

flag=1;

break;

}

i=0;

while((i<key)&&(flag==0))

{

if(a[i]==-1)

{

a[i]=getemp(emp,key);

flag=1;

break;

}

i++;

}

}

}

void main()

{

int num,key,i;

int ans=1;

system("clear");

printf("\n collision handling by linear probing:");

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

{

a[i]=-1;

}

do

{

printf("\n enter the four digit number:");

scanf("%d",&num);

key=create(num);

linear\_prob(key);

printf("\n Do you wish to continue?(1/0):");

scanf("%d",&ans);

}

while(ans);

display(emp);

}