Slip 1.

Que.1.

Implement a list library (doublylist.h) for a doubly linked list of integers

with the create, display operations. Write a menu driven program to call

these operations.

```
//doubly list header
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node{
 int data;
 struct node *next,*prev;
}*head,*temp,*newn;
void create()
{
 head==NULL;
 char ch[3];
 while(1){
 struct node *temp;
 newn=(struct node*)malloc(sizeof(struct node));

printf("\nEnter the data - ");

scanf("%d",&newn->data);
 newn->next=newn->prev=NULL;

if(head==NULL){

head=temp=newn;
 }
 else{
 temp->next=newn;
 newn->prev=temp;
 temp=temp->next;
 }
 printf("\nDo you want to enter more elements? - ");

scanf("%s",ch);

if(strcmp(ch,"n")==0)
 break;
 }
}
void display()
{
 struct node *temp;
 temp=head;

printf("\nDoubly linked list is - ");

while(temp!=NULL){

printf("%d\t",temp->data);
 temp=temp->next;
 }
}
//doubly linked list
#include<stdio.h>
#include<stdlib.h>
#include "1.1-doublylist.h"
int main()
{
 int ch;
 while(1){

printf("\n\nDOUBLY LINEAR LINKED LIST");
```

```
printf("\n1.Create a linked list");

printf("\n2.Display the list");

printf("\n3.Exit");

printf("\nEnter the choice from above - ");

scanf("%d",&ch);
 switch(ch){
 case 1 : create();
 break;
 case 2 : display();
 break;
 case 3 : exit(0);
 default: printf("\nEnter correct choice");
 }
 }
}
```

Slip 1

Que.2

Write a program that sorts the elements of linked list using any of sorting technique

```
//sort linked list
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node{
 int data;
 struct node *next;
}*head,*newn,*temp;
void create()
{
 char ch[3];
 head==NULL;
 while(1){
 struct node *temp;
 newn=(struct node*)malloc(sizeof(struct node));

printf("\nEnter the data - ");

scanf("%d",&newn->data);
 newn->next=NULL;

if(head==NULL){

head=temp=newn;
 }
 else{
 temp->next=newn;
 temp=temp->next;
 }
 printf("\nDo you want to enter more elements? ");

scanf("%s",ch);

if(strcmp(ch,"n")==0)
 break;
 }
}
void display()
{
 struct node *temp;
 temp=head;

printf("\nLinked list is - ");
```

```
while(temp!=NULL){

printf("%d\t",temp->data);
 temp=temp->next;
 }
}
void sort()
{
 struct node *i=head,*j=NULL;
 int curr;

if(head==NULL){
 return;
 }
 else
 {

while(i!=NULL)
 {
 j=i->next;


while(j!=NULL)
 {
 if(i->data>j->data)
 {
 curr=i->data;
 i->data=j->data;
 j->data=curr;
 }
 j=j->next;
 }
 i=i->next;
 }
 }
}
int main()
{
 int ch;
 while(1)
 {

printf("\nOperations on singly linked list - ");

printf("\n1.Create a linked list");

printf("\n2.Sort the list");

printf("\n3.Display list");

printf("\n4.Exit");

printf("\nEnter your choice - ");

scanf("%d",&ch);
 switch(ch)
 {
 case 1 : create();
 break;
 case 2 : sort();
 break;
 case 3 : display();
 break;
 case 4 : exit(0);
 default: printf("\nEnter correct choice - ");
 }
 }
}
-
```

Slip 2.

Que.1

Implement a list library (singlylist.h) for a singly linked list of integer

with the operations create, display. Write a menu driven program to call

these operations

```
//sigly linked list
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct node{
 int data;
 struct node *next;
}*head,*newn,*temp;
void create()
{
 char ch[3];
 head==NULL;

 while(1)
 {
 struct node *temp;
 newn=(struct node*)malloc(sizeof(struct node));

printf("\nEnter the data - ");

scanf("%d",&newn->data);
 newn->next=NULL;

if(head==NULL)
 {
head=temp=newn;
 }
 else
 {
 temp->next=newn;
 temp=newn;
 }
 printf("\nDo you want to enter more elements? - ");

scanf("%s",ch);

if(strcmp(ch,"n")==0)
 break;
 }
 printf("\n");
}
void display()
{
 struct node *temp;
 temp=head;

printf("\nNode is - ");

while(temp!=NULL){

printf("%d\t",temp->data);
 temp=temp->next;
 }
}
//singly linked list
#include<stdio.h>
#include<stdlib.h>
#include "2-1-singly.h"
int main()
{
 int ch;
```

```
while(1){

printf("\n\nSINGLY LINEAR LINKED LIST");

printf("\n1.Create a linked list");

printf("\n2.Display the list");

printf("\n3.Exit");

printf("\nEnter the choice from above - ");

scanf("%d",&ch);
switch(ch){
case 1 :
create();
break;
case 2 :
display();
break;
case 3 :
exit(0);
default:
printf("\nEnter correct choice");
}
}
}
```

Que.2

. Write a program that copies the contents of one stack into another. Use

stack library to perform basic stack operations. The order of two stacks

must be identical.(Hint: Use a temporary stack to preserve the order)

```
//evaluate postflix
#include<stdio.h>
int stack[20];
int top = -1;
void push(int x)
{
 stack[++top] = x;
}
int pop()
{
 return stack[top--];
}
int main()
{
 char exp[20];
 char *e;
 int n1,n2,n3,num;//for storing pop element
 printf("Enter the expression :: ");

scanf("%s",exp);
 e = exp;
 while(*e != '\0')
 {
 if(isdigit(*e))
 {
 num = *e - 48;//num for store value of epression
 push(num);
 }
 else
 {
 n1 = pop();//if e is operator then pop the two elemets from the
```

stack and sore it in var

```
n2 = pop();

switch(*e)//check the given operator and perform the operations accordingly
{
case '+':
{
n3 = n1 + n2;
break;
}
case '-':
{
n3 = n2 - n1;
break;
}
case '*':
{
n3 = n1 * n2;
break;
}
case '/':
{
n3 = n2 / n1;
break;
}
}
push(n3);
}
e++;
}
printf("\nThe result of expression %s = %d\n\n",exp,pop());
return 0;
```

Slip 3.

Que.1

Sort a random array of n integers (accept the value of n from user) in ascending order by using insertion sort algorithm.

```
// insertion sort using function
#include<stdio.h>
int insertion(int array[],int size)
{
for(int i=1;i<size;i++)
{
int temp=array[i];
int j=i-1;
while(j>=0 && array[j]>temp)
{
array[j+1]=array[j];
j--;
}
array[j+1]=temp;
}
}
int main()
{
int a[100],n;

printf("\nEnter size of array - ");

scanf("%d",&n);

printf("\nEnter %d elements in array - ",n);
for(int i=0;i<n;i++)
{
```

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

printf("\nUnsorted array is - ");
 for(int i=0;i<n;i++)
 {

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

insertion(a,n);

printf("\nSorted array is - ");
 for(int i=0;i<n;i++)
 {

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

Slip 4

Que.1

Read the ‘n’ numbers from user and sort using bubble sort.

```
//bubble sort using function
#include<stdio.h>
int bubble(int array[],int size)
{
 int i,j,temp;

for(i=0;i<size;i++)
 {
 for(j=0;j<size-i-1;j++)
 {

if(array[i]>array[i+1])
 {

temp=array[i];

array[i]=array[i+1];

array[i+1]=temp;
 }
 }
 }
 return 0;
}
int main()
{
 int n;

printf("\nHow many elements you want to enter? - ");

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

printf("\nEnter elements in the array:- ");

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

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

printf("\nUnsorted array is:- ");

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

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

printf("\nSort
```

ed array is:-
");

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

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

Slip 5

Que.1

Create a random array of n integers. Accept a value x from user and use

linear search algorithm to check whether the number is present in the

array or not and output the position if the number is present.

// linear search using functon

#include<stdi
o.h>

int linear(int array[],int size,int key)

{

//int flag;

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

{

if(array[i]==ke
y)

{

return 1;

}

}

return 0;

}

int main()

{

int a[5]={23,56,7
8,93,90};

int key;

printf("\n enter key to the search:-
");

scanf("%d",&
key);

int flag=linear(a,
5,key);

if(flag)

{

printf("\n key is found");

}

else

{

printf("\n key is not found");

}

}

Slip 5

Slip 5

Que.2

. Implement a priority queue library (PriorityQ.h) of integers using a static

implementation of the queue and

implement the below two operations.

1) Add an element with its priority into the queue.

2) Delete an element from queue according to its priorit

/*

* C Program to Implement Priority Queue to Add and Delete Elements

*/

#include
<stdio.h>

#include
<stdlib.h>

#define MAX
5

void
insert_by_pri
ority(int);

void
delete_by_pri
ority(int);

```
void create();
void check(int);
void display_pqueue();
int pri_que[MAX];
int front, rear;
void main()
{
 int n, ch;
 printf("\n1 - Insert an element into queue");
 printf("\n2 - Delete an element from queue");
 printf("\n3 - Display queue elements");
 printf("\n4 - Exit");
 create();
 while (1)
 {

printf("\nEnter your choice : ");
 scanf("%d", &ch);
 switch (ch)
 {
 case 1:

printf("\nEnter value to be inserted : ");

scanf("%d",&n);

insert_by_priority(n);
 break;
 case 2:

printf("\nEnter value to delete : ");

scanf("%d",&n);

delete_by_priority(n);
 break;
 case 3:

display_pqueue();
 break;
 case 4:
 exit(0);
 default:

printf("\nChoice is incorrect, Enter a correct choice");
 }
 }
}
/* Function to create an empty priority queue */
void create()
{
 front = rear = -1;
}
/* Function to insert value into priority queue */
void insert_by_priority(int data)
{
 if (rear >= MAX - 1)
 {

printf("\nQueue overflow no more elements can be inserted");
 return;
 }
```

```
 if ((front == -1) && (rear == -1))
 {
 front++;
 rear++;
 pri_que[rear] = data;
 return;
 }
 else
 check(data);
 rear++;
}
/* Function to check priority and place element */
void check(int data)
{
 int i,j;
 for (i = 0; i <= rear; i++)
 {
 if (data >= pri_que[i])
 {
 for (j = rear + 1; j > i; j--)
 {
 pri_que[j] = pri_que[j - 1];
 }
 pri_que[i] = data;
 return;
 }
 }
 pri_que[i] = data;
}
/* Function to delete an element from queue */
void delete_by_priority(int data)
{
 int i;
 if ((front==-1) && (rear==-1))
 {

printf("\nQueue is empty no elements to delete");
 return;
 }
 for (i = 0; i <= rear; i++)
 {
 if (data == pri_que[i])
 {
 for (; i < rear; i++)
 {
 pri_que[i] = pri_que[i + 1];
 }
 pri_que[i] = -99;
 rear--;
 if (rear == -1)
 front = -1;
 return;
 }
 }
 printf("\n%d not found in queue to delete", data);
}
/* Function to display queue elements */
void display_pqueue()
{
 if ((front == -1) && (rear == -1))
 {

printf("\nQueue is empty");
 return;
 }
```

```
 for (; front <= rear; front++)
 {
 printf(" %d ", pri_que[front]);
 }
 front = 0;
```

Slip 6.

Que.1

Sort a random array of n integers (accept the value of n from user) in ascending order by using selection sort algorithm.

```
//selection sort
#include<stdio.h>
int main()
{
 int n;
 printf("enter size of array : \n");

scanf("%d",&n);
 //create array
 int a[n],i;
 printf("enter %d elements in array : \n",n);

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

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

printf("Unsorted array is : \n");
 //display array

for(i=0;i<n;i++)
 {
 printf("%d \t",a[i]);
 }
 printf("\n");
 //selection sort begins
 int j,position,swap;
 for(i=0;i<n-1;i++)
 {
 position = i;

for(j=i+1;j<n;j++)
 {
 if(a[position] > a[j])
 {
 position = j;
 }
 }
 swap = a[i];
 a[i] = a[position];
 a[position] = swap;
 }

printf("sorted array is : \n");
 //selection sorted array is

for(i=0;i<n;i++)
 {
 printf("%d \t",a[i]);
 }
 //sorting end
}
```

Slip 6

Que.2

Implement a queue library (dyqueue.h) of integers using a dynamic

(linked list) implementation of the queue and implement init, enqueue,

dequeue, isempty, peek operations.

```
//dynamic insertion of queue
#include<stdio.h>
#include"6-2-dyqueue.h"
int main()
{
 int n,ch;
 //clrscr();
 initqueue();
 do
 {

printf("\n\n1.ADD\n2.DELETE\n3.DISPLAY\n4.PEEK\n5.EXIT");

printf("\nEnter your choice - ");

scanf("%d",&ch);
 switch(ch)
 {
 case 1 : printf("\nEnter the data - ");

scanf("%d",&n);
 enqueue(n);
break;
 case 2 : dequeue();
 break;
 case 3 : display();
 break;
 case 4 : peek();
 break;
 case 5 : exit(0);
 }
 } while (ch!=4);
 return 0;
}
//dynamic insertion of queue
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAX 3
struct queue
{
 int data[MAX];
 int front,rear;
}q;
//typedef struct q;
void initqueue()
{
 q.front=0;
 q.rear=-1;
}
int enqueue(int num)
{

if(q.rear!=MAX-1)
 {
 ++q.rear;

q.data[q.rear]=num;
 printf("%d is added at %d position",num,q.rear);
 }
 else

printf("\nQueue is full!!!");
```

```
}
int dequeue()
{
 if(q.front==-1 && q.rear==-1)

printf("\nQueue is empty!!!");
 else if(q.front>q.rear){

q.rear=q.front=-1;

printf("\nQueue is empty!!!");
 }
 else if(q.rear!=-1){
 int x=q.data[q.front];
 q.front++;

printf("\nDeleted data is %d",x);
 }
}
void display()
{
 int i;

 printf("\nFront -> %d",q.front);

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

printf("\nData -> %d\t",q.data[i]);
 printf("\nRear -> %d",q.rear);

if(q.front>q.rear || q.front==-1 && q.rear==-1)

printf("\nQueue is empty!!!");
}
void peek()
{

printf("\nPeek element is %d",q.data[q.front]);}
```

Slip 7.

Que.1

Sort a random array of n integers (accept the value of n from user) in ascending order by using quick sort algorithm.

```
//quick sort
#include<stdio.h>
int partition(int *a,int lb,int ub) //function defination
{
 //Entered in Partioned
 int up,down,pivot,swap;

 down = lb;
 up = ub;
 pivot = a[lb];
 do{

while((a[down]<=pivot) && (down < ub))
 {
 down ++;
 }

while(a[up]>pivot && up> lb)
```

```
{
up--;
}

if(down < up)
{
swap = a[down];
a[down] = a[up];
a[up] = swap;
}
}while(down < up);
a[lb] = a[up];
a[up] = pivot;
return up;
}
void quicksort(int *a,int lb,int ub) //function defination
{
//Entered in quick sort
int j;
if(lb < ub)
{
//calling function
j = partition(a,lb, ub);

quicksort(a,lb ,j-1);

quicksort(a,j+ 1, ub);
}
}
int main()
{
int n;
printf("enter the size of array : \n");

scanf("%d",& n);
int a[n],i;
//create array
printf("enter %d elements in array \n",n);

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

scanf("%d",& a[i]);
}
//calling function

quicksort(a,0, n-1);

//sorted array
printf("Sortd array is : \n");

for(i=0;i<n;i++ )
{
printf("%d \t",a[i]);
}
printf("\n");
}
```

Slip 7

Que.2

. Write a program that checks whether a string of characters is palindrome or not. The function should use a stack library (cststack.h) of stack of characters using a static implementation of the stack

```
//palindrome( static stack implimentation)
#include<stdio.h>
```

```
#include<stdlib.h>
#include "7-2-palindrome.h"
void palindrome()
{
 char string[15];
 int i,count=0,len;

printf("\nEnter a string:");

scanf("%s",string);

len=strlen(string);

for(i=0;i<len;i++)
 {

push(string[i]);
 }
 for (i = 0; i <len; i++)
 {
 if (string[i] == pop())
 count++;
 }

 if (count == len)
 printf("%s is a Palindrome string\n", string);
 else
 printf("%s is not a palindrome string\n", string);
}
int main()
{
 int ch,i;
 char string[15];
 while(1)
 {

printf("\nCHECK PALINDROME");

printf("\n1.Init\n");

printf("2.Check Palindrome String\n");

printf("3.IsEmpty\n");

printf("4.Exit\n");
 printf("Enter your choice:");

scanf("%d",&ch);
 switch(ch)
 {
 case 1:init();break;
 case 2: palindrome();break;
 case 3:isempty();break;
 case 4:exit(0);
 }
 }
}
//palindrome
#include<stdio.h>
#include<string.h>
#define MAX 15
char name[MAX],top;
void init()
{
 top=-1;
```

```
printf("\nStatic Stack Initiaized\n");
}
void push(char c)
{
 top++;
 name[top]=c;
}
char pop()
{
 return name[top--];
}
isempty()
{
 if(top==-1)
 {

printf("\nStatic Stack is empty\n");
 }
 else
 {

printf("\nStatic Stack is not empty\n");
 }
```

Slip 8.

Que.1

. Implement a list library (singlylist.h) for a singly linked list of integer

With the operations create, delete specific element and display. Write a

menu driven program to call these operation

```
//singly list
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct node{
 int data;
 struct node *next;
}*head,*newn,*temp;
void create()
{
 char ch[3];
 head==NULL;

 while(1)
 {
 struct node *temp;
 newn=(struct node*)malloc(sizeof(struct node));

printf("\nEnter the data - ");

scanf("%d",&newn->data);
 newn->next=NULL;

if(head==NULL)
 {

head=temp=newn;
 }
 else
 {
 temp->next=newn;
 temp=newn;
 }
 printf("\nDo you want to enter more elements? - ");

scanf("%s",ch);

if(strcmp(ch,"n")==0)
 break;
 }
```

```
 printf("\n");
}
void delete()
{
 temp=head;
 int pos,i;
 struct node *curr;

printf("\nEnter the position to delete - ");

scanf("%d",&pos);
 if(pos==1)
 {
 head=head->next;
 free(temp);
 }
 else
 {
 i=1;
 temp=head;
 }
 while((i<pos-1)&&(temp->next!=NULL))
 {
 i++;
 temp=temp->next;
 }
 if(temp->next==NULL)

printf("\nNodes not present");
 else
 {
 curr=temp->next;
 temp->next=curr->next;
 free(curr);
 }
}
void display()
{
 temp=head;

printf("\nElements of list are - ");

while(temp!=NULL)
 {

printf("%d\t",temp->data);
 temp=temp->next;
 }
 printf("\n");
}
//singly list
#include<stdio.h>
#include<stdlib.h>
#include"7-1-singly.h"
int main()
{
 int ch;
 while(1)
 {

printf("\nOperations on singly linked list - ");

printf("\n1.Create a linked list");

printf("\n2.Delete element from the list");

printf("\n3.Display elements of list");

printf("\n4Exit");

printf("\nEnter your choice - ");
```

```
scanf("%d",&ch);
 switch(ch)
 {
 case 1 : create();
 break;
 case 2 : delete();
 break;
 case 3 : display();
 break;
 case 4 : exit(0);
 default: printf("\nEnter correct choice - ");
 }
 }
}
```

Slip 9.

Que.1

. Write a program to convert an infix expression of the form (a*(b+c)*((da)/b)) into its equivalent postfix notation. Consider usual precedence's of operators. Use stack library of stack of characters using static implementation.

```
//inflix to postflix
#include<stdio.h>
#include<ctype.h>
char stack[100];
int top=-1;
void push(char x){

stack[++top]=x;
}
char pop()
{
 if(top==-1)
 return -1;
 else
 return stack[top--];
}
int priority(char x){
 if(x=='(')
 return 0;

if(x=='+'||x=='-')
 return 1;

if(x=='*'||x=='/')
 return 2;
 return 0;
}
int main()
{
 char exp[100];
 char *e,x;
 printf("Enter the expression - ");

scanf("%s",exp);
 printf("\n");
 e=exp;

while(*e!='\0'){

if(isalnum(*e))

printf("%c",*e);
 else if(*e=='(')
 push(*e);
 else if(*e==')')
 {

while((x=pop())!='(')
```

```
printf("%c",x);
 }
 else
 {

while(priority
(stack[top])>=
priority(*e))

printf("%c",p
op());
 push(*e);
 }
 e++;
 }
 while(top!=-
1)
 {
 printf("%c
",pop());
 }
 return 0;
}
```

Slip 9

Que.2

Read the data from the 'employee.txt' file and sort on age using Counting sort or Quick sort and write the sorted data to another file 'sortedemponage.txt'

```
//read the
data from
emp_quick_a
ge.txt and
sort bt age
#include<stdi
o.h>
#include<strin
g.h>
typedef struct
{
 char
name[30];
 int age;
 int salary;
}
RECORD;
RECORD
emp[100];
int
readfile(RECO
RD[]);
void
writefile(REC
ORD[],int);
int
quicksort(REC
ORD *emp,int
low,int high);
int
partition(REC
ORD *emp,
int low, int
high);
void
sort(RECORD
*emp,int n);
int main()
{
 int n;

n=readfile(em
p);
 sort(emp,n);
writefile(emp,
n);
}
int
readfile(RECO
RD *a)
{
 int i=0;
 FILE *fp;
 if
((fp=fopen("e
mp_quick_ag
e.txt","r"))!=N
ULL)
 while(!
feof(fp))
 {

fscanf(fp,"%s
%d%d",a[i].na
me,&a[i].age,
&a[i].salary);
 i++;
 }
 return i;
}
```

```
void writefile(RECORD *a,int n)
{
 int i=0;
 FILE * fp;

if((fp=fopen("sorted_emp_quick_age.txt","w"))!=NULL)

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

fprintf(fp,"%s\t%d\t%d\n",a[i].name,a[i].age,a[i].salary);
}
int quicksort(RECORD *emp,int low,int high)
{
 if( low < high )
 {
 int p;
 p = partition(emp, low, high);

quicksort(emp, low, p-1);

quicksort(emp, p+1, high);
}
}
void sort(RECORD *emp, int n)
{

quicksort(emp,0,n-1);
}
int partition(RECORD *emp, int low, int high)
{
 RECORD pivot = emp[low];
 int start, end;
 start = low;
 end = high;
 while( start < end )
 {
 while( start <= high && emp[start].age <= pivot.age )
 start++;
 while( emp[end].age > pivot.age )
 end--;
 if( start < end )
 {
 RECORD swap = emp[start];
 emp[start] = emp[end];
 emp[end] = swap;
 }
 }
 emp[low] = emp[end];
 emp[end] = pivot;
 return end;
}
//make a file emp_quick_age.txt and list some name age and salary of emp
//wiev the orderd age wise result in sorted_ emp_quick_age.txt file
```

Slip.10

Que.2

. Read the data from the file “employee.txt” and sort on names in

alphabetical order (use strcmp) using bubble sort or selection sort

//read the data from txt file and sort the names using bubble sort

```
#include<stdio.h>
#include<string.h>
typedef struct
{
char name[30];
int age;
int salary;
}
RECORD;
RECORD emp[100];
int readfile(RECORD[]);
void writefile(RECORD[],int);
int sort(RECORD *emp,int n);
int main()
{
 int n;

n=readfile(emp);
 sort(emp,n);

writefile(emp,n);
}
int readfile(RECORD *a)
{
 int i=0;
 FILE *fp;
 if ((fp=fopen("10-2emp_bubble.txt","r"))!=NULL)
 while(!feof(fp))
 {

fscanf(fp,"%s %d%d",a[i].name,&a[i].age,&a[i].salary);
 i++;
 }
 return i;
}
void writefile(RECORD *a,int n)
{
 int i=0;
 FILE * fp;

if((fp=fopen("10-2sortedemp_bubble.txt","w"))!=NULL)

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

fprintf(fp,"%s\t%d\t%d\n",a[i].name,a[i].age,a[i].salary);
}

 void swap(RECORD *a, RECORD *b)
 {
 RECORD tmp=*a;
 *a=*b;
 *b=tmp;
```

```
}


int sort(RECORD * emp,int n)
{
int i,j,pass;

for(pass=1;pass<n;pass++)
{
for(i=0;i<n-pass;i++)
{
if (strcmp (emp[i].name, emp[i+1].name)>0)
{

swap(&emp[i],&emp[i+1]);
}
}
}
}
```

```
//make text file name as 10-2emp_bubble.txt save it in vs code
//and output is displayed in the 10-2sortedemp_bubble.txt file
```

Slip.11

Que.2

```
//priority queue
#include <stdio.h>
#include <stdlib.h>
#include "11-2priorityq.h"
void main()
{
int n, ch;
printf("\n1 - Insert an element into queue");
printf("\n2 - Delete an element from queue");
printf("\n3 - Display queue elements");
printf("\n4 - Exit");
create();
while (1)
{

printf("\nEnter your choice : ");
scanf("%d", &ch);
switch (ch)
{
case 1:

printf("\nEnter value to be inserted : ");

scanf("%d",&n);
insert_by_priority(n);
break;
case 2:

printf("\nEnter value to delete : ");

scanf("%d",&n);

delete_by_priority(n);
break;
case 3:

display_pqueue();
break;
case 4:
exit(0);
default:

printf("\nChoice is incorrect,
```

```
Enter a correct choice");
 }
 }
}
//priority queue
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int pri_que[MAX];
int front, rear;
/* Function to create an empty priority queue */
void create()
{
 front = rear = -1;
}
/* Function to insert value into priority queue */
void insert_by_priority(int data)
{
 if (rear >= MAX - 1)
 {

printf("\nQueue overflow no more elements can be inserted");
 return;
 }
 if ((front == -1) && (rear == -1))
 {
 front++;
 rear++;
 pri_que[rear] = data;
 return;
 }
 else
 check(data);
 rear++;
}
/* Function to check priority and place element */
void check(int data)
{
 int i,j;
 for (i = 0; i <= rear; i++)
 {
 if (data >= pri_que[i])
 {
 for (j = rear + 1; j > i; j--)
 {
 pri_que[j] = pri_que[j - 1];
 }
 pri_que[i] = data;
 return;
 }
 }
 pri_que[i] = data;
}
/* Function to delete an element from queue */
void delete_by_priority(int data)
{
 int i;
 if ((front==-1) && (rear==-1))
 {

printf("\nQueue is empty no elements to delete");
 return;
```

```
}
for (i = 0; i <= rear; i++)
{
if (data == pri_que[i])
{
for (; i < rear; i++)
{
pri_que[i] = pri_que[i + 1];
}
pri_que[i] = -99;
rear--;
if (rear == -1)
front = -1;
return;
}
}
printf("\n%d not found in queue to delete", data);
}
/* Function to display queue elements */
void display_pqueue()
{
if ((front == -1) && (rear == -1))
{

printf("\nQueue is empty");
return;
}
for (; front <= rear; front++)
{
printf(" %d ", pri_que[front]);
}
front = 0;
}
```

Slip.12

Que.1

```
//read data from 12-2-cities.txt file and apply linear search to find cities
#include<stdio.h>
#include<string.h>
typedef struct city
{
char name[20];
int code;
} city;
//Fileread
int readfile(city a[20])
{
FILE *fp;
int i=0;

fp=fopen("12-2-cities.txt","r");
if(fp==NULL)
printf("File Not Exist");
else
{

while(!feof(fp))
{

fscanf(fp,"%s %d", a[i].name, &a[i].code);
i++;
}
fclose(fp);
}
return i;
}
int main()
{
```

```
 int i, n;
 char key[20];
 city a[20];
 n = fileread(a);
 for(int i=0; i<n; i++)
 printf("%s %d\n", a[i].name, a[i].code);
 /* displaying records*/

linearsearch(n);
}
//Linear Search
int linearsearch(int n)
{
city a[20];
 n=fileread(a);
 char str[20];
 int index,flag=0;
 printf("Enter city:");

scanf("%s",str);
 for(int i=0;i<n;i++)
 {

if(strcmp(str,a[i].name)==0)
 {
 flag=1;
 index=i;
 }
 }
 if(flag==1)
 printf("City Code: %d",a[index].code);
 else
 printf("City Not in list");
}
```

Slip.13

Que.2

```
//sort list using bubble sort
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node{
 int data;
 struct node *next;
}*head,*newn,*temp;
void create()
{
 char ch[3];
 head=NULL;
 while(1){
 struct node *temp;
 newn=(struct node*)malloc (sizeof(struct node));

printf("\nEnter the data - ");

scanf("%d",&newn->data);
 newn->next=NULL;

if(head==NULL){

head=temp=newn;
 }
 else{
 temp->next=newn;
 temp=newn;
 }
 printf("\nDo you want to enter more elements? ");
```

```
scanf("%s",ch
);

if(strcmp(ch,"
n")==0)
 break;
 }
}
void display()
{
 struct node
*temp;
 temp=head;
 printf("\nList
is - ");

while(temp!=
NULL){

printf("%d\t",
temp->data);
 temp=temp-
>next;
 }
}
void sort()
{
 struct node
*i,*j;
 int curr;
 for(i=head;i-
>next!=NULL;i
=i->next){
 for(j=i-
>next;j!=NULL
;j=j->next){
 if(i->data>j-
>data){
 curr=i->data;
 i->data=j-
>data;
 j->data=curr;
 }
 }
 }
}
int main()
{
 int ch;
 while(1){

printf("\nSIN
GLY LINKED
LIST");

printf("\n1.Cr
eate the list");

printf("\n2.So
rt the list");

printf("\n3.Di
splay the
list");

printf("\n4.Ex
it");

printf("\nEnte
r the choice
from the
above - ");

scanf("%d",&
ch);
 switch(ch){
 case 1 :
create();
 break;
 case 2 :
sort();
 break;
 case 3 :
display();
 break;
 case 4 :
exit(0);
 default:
printf("\nEnte
r correct
choice");
 }
 }
}
```

Slip.14

Que.1

```
//linear search
#include<stdio.h>
int linear(int array[], int size, int key)
{
 int i;

for(i=0;i<size;i++){

if(key==array[i]){
 return 1;
 }
 }
 return 0;
}
int main()
{
 int a[100],n,i,key;

printf("\nEnter the size of array - ");

scanf("%d",&n);
 a[n];
 printf("Enter elements in array - ");

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

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

printf("\nGiven array is - ");

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

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

printf("\nEnter key to the search - ");

scanf("%d",&key);
 int found=linear(a,n,key);

if(key==found){
 printf("\nkey is found at %d location",i);
 }
 else{
 printf("\nKey is not found!!!");
 }
```

Slip.15

Que.1

```
//selection sort
#include<stdio.h>
int main()
{
 int n;
 printf("enter size of array : \n");

scanf("%d",&n);
 //create array
 int a[n],i;
 printf("enter %d elements in array : \n",n);

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

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

printf("Unsorted array is :\n");
 //display array

for(i=0;i<n;i++)
 {
 printf("%d \t",a[i]);
 }
 printf("\n");
 //selection sort begins
 int j,position,swap;
 for(i=0;i<n-1;i++)
 {
 position = i;
for(j=i+1;j<n;j++)
 {
 if(a[position] > a[j])
 {
 position = j;
 }
 }
 swap = a[i];
 a[i] = a[position];
 a[position] = swap;
 }

printf("sorted array is : \n");
 //selection sorted array is

for(i=0;i<n;i++)
 {
 printf("%d \t",a[i]);
 }
 //sorting end
}
```

Slip.17

Que.1

```
//singly reverse
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
 int data;
 struct node *next;
}*head,*newn,*temp;
void create();
void display();
void reverse();
int main()
{
 int ch;
 while(1)
 {

printf("\nOperations on singly linked list - ");

printf("\n1.Create a linked list");

printf("\n2.Display elements of list");

printf("\n3.Reverse a list");

printf("\n4.Exit");

printf("\nEnter your choice - ");
```

```
scanf("%d",&ch);
 switch(ch)
 {
 case 1 : create();
 break;
 case 2 : display();
 break;
 case 3 : reverse();
 break;
 case 4 : exit(0);
 default: printf("\nEnter correct choice - ");
 }
 }
}
void create()
{
 char ch[3];
 head==NULL;

 while(1)
 {
 struct node *temp;
 newn=(struct node*)malloc(sizeof(struct node));

printf("\nEnter the data - ");

scanf("%d",&newn->data);
 newn->next=NULL;

if(head==NULL)
 {

head=temp=newn;
 }
 else
 {
 temp->next=newn;
 temp=newn;
 }
 printf("\nDo you want to enter more elements? - ");

scanf("%s",ch);

if(strcmp(ch,"n")==0)
 break;
 }
 printf("\n");
}
void display()
{
 temp=head;

printf("\nElements of list are - ");

while(temp!=NULL)
 {

printf("%d\t",temp->data);
 temp=temp->next;
 }
 printf("\n");
}
void reverse()
{
 struct node *t1,*t2;
 t1=t2=NULL;

while(head!=NULL)
 {
 t2=head->next;
 head->next=t1;
 t1=head;
 head=t2;
```

```
 }
 head=t1;
}
```

Slip 17

Que.2

```
//copy stack
#include<stdlib.h>
#include"2-2-copystack.h"
void main()
{
 stack orginal,temp,c_or;
 int n,x;

init(&orginal);
 init(&temp);
 init(&c_or);
 printf("Enter no of nodes\n");

scanf("%d",&n);
 printf("Enter %d elements\n",n);
 for(int i=0;i<n;i++){

scanf("%d",&x);
//accpeting orignal stack...

push(&orginal,x);
 }
 for(int i=0;i<n;i++)

push(&temp,pop(&orginal)); //copy orignal stack into temp stack

printf("\npeek element of temp stack is %d",peek(temp));
 for(int i=0;i<n;i++)

push(&c_or,pop(&temp));//to copy temp stack into c_stack;

printf("\npeek element of temp stack is %d ",peek(c_or));
 }
//copy stack
#include<stdio.h>
#define MAX 20
typedef struct
{
 int top;
 int st[MAX];
}stack;

int peek(stack a)
{
 return a.st[a.top];
}
void init(stack * a)
{
 a->top=-1;
}
void push(stack * a,int x)
{
 if(a->top==MAX-1)
printf("\nstack is full ");
 else{
 a->st[++a->top]=x;
 }
}
int pop(stack *a)
```

```
{
 if(a->top>=MAX-1) return 0;
 else
 return a->st[a->top--];
}
```

Slip.19

Que.1

```
//selection sort
#include<stdio.h>
int main()
{
 int n;
 printf("enter size of array : \n");

scanf("%d",&n);
 //create array
 int a[n],i;
 printf("enter %d elements in array : \n",n);

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

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

printf("Unsorted array is : \n");
 //display array

for(i=0;i<n;i++)
 {
 printf("%d \t",a[i]);
 }
 printf("\n");
 //selection sort begins
 int j,position,swap;
 for(i=0;i<n-1;i++)
 {
 position = i;

for(j=i+1;j<n;j++)
 {
 if(a[position] > a[j])
 {
 position = j;
 }
 }
 swap = a[i];
 a[i] = a[position];
 a[position] = swap;
 }

printf("sorted array is : \n");
 //selection sorted array is

for(i=0;i<n;i++)
 {
 printf("%d \t",a[i]);
 }
 //sorting end
}
```

Slip.20

Que.1

```
//Array as a stack(static)
#include<stdio.h>
#include<stdlib.h>
#include "20-1static_stack.h"
int main()
{
 int ch;
 while(1)
 {

printf("\nEnter a number to perform operations on stack : ");

printf("\n1.Push");

printf("\n2.Pop");

printf("\n3.Display");

printf("\n4.Peek");

printf("\n5.Exit");

printf("\nEnter your choice : ");

scanf("%d",&ch);

 switch(ch)
 {
 case 1 : push();
 break;
 case 2 : pop();
 break;
 case 3 : display();
 break;
 case 4 : printf("\nTop most element of stack is %d",peek());
 break;
 case 5 : exit(0);
 default: printf("\nEnter the correct choice : ");
 }
 }
}
#include<stdio.h>
#include<stdlib.h>
#define MAX 3
int top=-1,stack[MAX];
void push()
{
 int data;

printf("\nEnter the data to push - ");

scanf("%d",&data);
 if(isfull())
 {

printf("\nStack Overflow");
 }
 top+=1;

stack[top]=data;
}
int pop()
{
 int p;
 if(isempty())
 {

printf("\nStack Underflow");
```

```
}

p=stack[top];
top=top-1;

printf("\nPoped element is %d",p);
}
int peek(){

if(top==-1)
{

printf("\nStack Underflow");
exit(1);
}
return stack[top];
}
void display()
{
int i;
if(top==-1)

printf("\nStack Underflow");
else
{

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

printf("stack[%d]=%d\n",i,stack[i]);
}
}
int isfull()
{
if(top==MAX-1)
return 1;
else
return 0;
}
int isempty()
{
if(top==-1)
return 1;
else
return 0;
```

Slip.21

Que.1

```
//reverse string
#include<stdio.h>
#include<string.h>
#define MAX 100
//maximum no of character
int top=-1;
int item;
char stack_string[MAX];
void pushchar(char item)
{
if(isfull())
{
printf("\n stack is full \n");
return;
}
top+=1;

stack_string[top]=item;
}
char popchar()
{
if(isempty())
{
printf("\n stack is empty \n");
return 0;
}
item = stack_string[top];
top-=1;
return item;
}
```

```
int isempty()
{
 if(top==-1)
 return 1;
 else
 return 0;
}
int isfull()
{
 if(top==MAX-1)
 return 1;
 else
 return 0;
}
//reverse string using array(static implimentation)
#include<stdio.h>
#include "21-1cstacks.h"
#define MAX 100
int main()
{
 char str[MAX];
 int i;
 printf("input a string :- ");

scanf("%[^\n]s",str); //read string with spaces

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

pushchar(str[i]);

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

str[i]=popchar();

printf("reversed string is :- %s \n",str);
 return 0;
}
```

Slip 21

Que.2

```
//read data from 21-2-emp_insertion_name.txt file and sort the names
#include<stdio.h>
#include<string.h>
typedef struct
{
char name[50];
int age;
int salary;
}
RECORD;
RECORD emp[200];
int readfile(RECORD[]);
void writefile(RECORD[],int);
int sort(RECORD *emp,int n);
int main()
{
 int n;

n=readfile(emp);
 sort(emp,n);

writefile(emp,n);
}
int readfile(RECORD *a)
{
 int i=0;
 FILE *fp;
 if ((fp=fopen("21-2-emp_insertio
```

```
n_name.txt","r"))!=NULL)
 while(!feof(fp))
 {

fscanf(fp,"%s %d%d",a[i].name,&a[i].age,&a[i].salary);
 i++;
 }
 return i;
}
void writefile(RECORD *a,int n)
{
 int i=0;
 FILE * fp;

if((fp=fopen("21-2-sorted_emp_insertion_name.txt","w"))!=NULL)

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

fprintf(fp,"%s\t%d\t%d\n",a[i].name,a[i].age,a[i].salary);
}
 void swap(RECORD *a, RECORD *b)
 {
 RECORD tmp=*a;
 *a=*b;
 *b=tmp;
 }
int sort(RECORD *emp,int n)
{
 RECORD temp;
 int i,j;

for(i=1;i<n;i++)
 {

temp=emp[i];
 for(j=i-1;j>=0 && strcmp(emp[j].name,temp.name)>0;j--)
 {

emp[j+1]=emp[j];
 }

emp[j+1]=temp;
 }
}
-
```

Slip.23

Que.1

```
//priority queue

#include <stdio.h>
#include <stdlib.h>
#include "23-1-priorityq.h"
void main()
{
 int n, ch;
 printf("\n1 - Insert an element into queue");
 printf("\n3 - Display queue elements");
 printf("\n4 - Exit");
 create();
 while (1)
 {
```

```
printf("\nEnter your choice : ");
 scanf("%d", &ch);
 switch (ch)
 {
 case 1:

printf("\nEnter value to be inserted : ");

scanf("%d",&n);

insert_by_priority(n);
 break;
 case 3:

display_pqueue();
 break;
 case 4:
 exit(0);
 default:

printf("\nChoice is incorrect, Enter a correct choice");
 }
 }
}
//priority queue
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
int pri_que[MAX];
int front, rear;
/* Function to create an empty priority queue */
void create()
{
 front = rear = -1;
}
/* Function to insert value into priority queue */
void insert_by_priority(int data)
{
 if (rear >= MAX - 1)
 {

printf("\nQueue overflow no more elements can be inserted");
 return;
 }
 if ((front == -1) && (rear == -1))
 {
 front++;
 rear++;
 pri_que[rear] = data;
 return;
 }
 else
 check(data);
 rear++;
}
/* Function to check priority and place element */
void check(int data)
{
 int i,j;
 for (i = 0; i <= rear; i++)
 {
 if (data >= pri_que[i])
 {
 for (j = rear + 1; j > i; j--)
```

```
{
 pri_que[j] = pri_que[j - 1];
}
 pri_que[i] = data;
 return;
}
}
 pri_que[i] = data;
}
/* Function to display queue elements */
void display_pqueue()
{
 if ((front == -1) && (rear == -1))
 {

printf("\nQueue is empty");
 return;
 }
 for (; front <= rear; front++)
 {
 printf(" %d ", pri_que[front]);
 }
 front = 0;
}
```

Slip.24

Que.2

```
//read data from 24-2-emp_insertion_name.txt file and sort the names
#include<stdio.h>
#include<string.h>
typedef struct
{
char name[50];
int age;
int salary;
}
RECORD;
RECORD emp[200];
int readfile(RECORD[]);
void writefile(RECORD[],int);
int sort(RECORD *emp,int n);
int main()
{
 int n;

n=readfile(emp);
 sort(emp,n);

writefile(emp,n);
}
int readfile(RECORD *a)
{
 int i=0;
 FILE *fp;
 if ((fp=fopen("24-2-emp_insertion_name.txt","r"))!=NULL)
 while(!feof(fp))
 {

fscanf(fp,"%s %d%d",a[i].name,&a[i].age,&a[i].salary);
 i++;
 }
 return i;
}
void writefile(RECORD *a,int n)
{
```

```
int i=0;
FILE * fp;

if((fp=fopen("24-2-sorted_emp_insertion_name.txt","w"))!=NULL)

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

fprintf(fp,"%s\t%d\t%d\n",a[i].name,a[i].age,a[i].salary);
}
void swap(RECORD *a, RECORD *b)
{
RECORD tmp=*a;
*a=*b;
*b=tmp;
}
int sort(RECORD *emp,int n)
{
RECORD temp;
int i,j;

for(i=1;i<n;i++)
{

temp=emp[i];
for(j=i-1;j>=0 && strcmp(emp[j].name,temp.name)>0;j--)
{

emp[j+1]=emp[j];
}

emp[j+1]=temp;
}
}
```

Slip.26

Que.1

```
//Array as a stack(static)
#include<stdio.h>
#include<stdlib.h>
#include "20-1static_stack.h"
int main()
{
int ch;
while(1)
{

printf("\nEnter a number to perform operations on stack : ");

printf("\n1.Push");
printf("\n2.Pop");

printf("\n3.Display");

printf("\n4.Peek");

printf("\n5.Exit");

printf("\nEnter your choice : ");

scanf("%d",&ch);

switch(ch)
{
case 1 : push();
break;
case 2 : pop();
break;
```

```
 case 3 :
display();
 break;
 case 4 :
printf("\nTop
most element
of stack is
%d",peek());
 break;
 case 5 :
exit(0);
 default:
printf("\nEnte
r the correct
choice : ");
 }
 }
}
#include<stdi
o.h>
#include<stdli
b.h>
#define MAX
3
int top=-
1,stack[MAX];
void push()
{
 int data;

printf("\nEnte
r the data to
push - ");

scanf("%d",&
data);
 if(isfull())
 {

printf("\nStac
k Overflow");
 }
 top+=1;

stack[top]=da
ta;
}
int pop()
{
 int p;
 if(isempty())
 {

printf("\nStac
k
Underflow");
 }

 p=stack[top];
 top=top-1;

printf("\nPop
ed element is
%d",p);
}
int peek(){

 if(top==-1)
 {

printf("\nStac
k
Underflow");
 exit(1);
 }
 return
stack[top];
}
void display()
{
 int i;
 if(top==-1)

printf("\nStac
k
Underflow");
 else
 {

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

printf("stack[
%d]=%d\n",i,s
tack[i]);
 }
}
int isfull()
{
 if(top==MAX-
1)
 return 1;
 else
 return 0;
}
int isempty()
```

```
{
 if(top==-1)
 return 1;
 else
 return 0;
}
```

Que.2

---------SF026-

Slip.27

Que.1

Que.2

---------SF026-

Slip.28

Que.1

Slip.26

Que.2

```
//read the data from emp_quick_age.txt and sort bt age
#include<stdio.h>
#include<string.h>
typedef struct
{
 char name[30];
 int age;
 int salary;
}
RECORD;
RECORD emp[100];
int readfile(RECORD[]);
void writefile(RECORD[],int);
int quicksort(RECORD *emp,int low,int high);
int partition(RECORD *emp, int low, int high);
void sort(RECORD *emp,int n);
int main()
{
 int n;

n=readfile(emp);
 sort(emp,n);

writefile(emp, n);
}
int readfile(RECORD *a)
{
 int i=0;
 FILE *fp;
 if ((fp=fopen("emp_quick_age.txt","r"))!=NULL)
 while(! feof(fp))
 {

fscanf(fp,"%s %d%d",a[i].name,&a[i].age, &a[i].salary);
 i++;
 }
 return i;
}
void writefile(RECORD *a,int n)
{
 int i=0;
 FILE * fp;

if((fp=fopen("sorted_emp_quick_age.txt
```

```
","w"))!=NULL
)

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

fprintf(fp,"%s
\t%d\t%d\n",
a[i].name,a[i].
age,a[i].salary
);
}
int
quicksort(REC
ORD *emp,int
low,int high)
{
 if( low < high
)
 {
 int p;
 p =
partition(emp
, low, high);

quicksort(em
p, low, p-1);
quicksort(em
p, p+1, high);
}
}
void
sort(RECORD
*emp, int n)
{

quicksort(em
p,0,n-1);
}
int
partition(REC
ORD *emp,
int low, int
high)
{
 RECORD
pivot =
emp[low];
 int start, end;
 start = low;
 end = high;
 while( start <
end )
 {
 while( start
<= high &&
emp[start].ag
e <= pivot.age
)
 start++;
 while(
emp[end].age
> pivot.age )
 end--;
 if( start < end
)
 {
 RECORD
swap =
emp[start];
 emp[start] =
emp[end];
 emp[end] =
swap;
 }
 }
 emp[low] =
emp[end];
 emp[end] =
pivot;
 return end;
}
```

Slip.29

Que.1

```
//Array as a
stack(static)
#include<stdi
o.h>
#include<stdli
b.h>
#include "20-
1static_stack.
h"
int main()
{
 int ch;
 while(1)
 {

printf("\nEnte
r a number to
perform
operations on
stack : ");

printf("\n1.Pu
sh");
```

```
printf("\n2.Pop");

printf("\n3.Display");

printf("\n4.Peek");

printf("\n5.Exit");

printf("\nEnter your choice : ");

scanf("%d",&ch);

 switch(ch)
 {
 case 1 : push();
 break;
 case 2 : pop();
 break;
 case 3 : display();
 break;
 case 4 : printf("\nTop most element of stack is %d",peek());
 break;
 case 5 : exit(0);
 default: printf("\nEnter the correct choice : ");
 }
 }
}

#include<stdio.h>
#include<stdlib.h>
#define MAX 3
int top=-1,stack[MAX];
void push()
{
 int data;

printf("\nEnter the data to push - ");

scanf("%d",&data);
 if(isfull())
 {

printf("\nStack Overflow");
 }
 top+=1;

stack[top]=data;
}
int pop()
{
 int p;
 if(isempty())
 {

printf("\nStack Underflow");
 }

 p=stack[top];
 top=top-1;

printf("\nPoped element is %d",p);
}
int peek(){

 if(top==-1)
 {

printf("\nStack Underflow");
 exit(1);
 }
 return stack[top];
}
void display()
{
```

```
 int i;
 if(top==-1)

printf("\nStac
k
Underflow");
 else
 {

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

printf("stack[
%d]=%d\n",i,s
tack[i]);
 }
}
int isfull()
{
 if(top==MAX-
1)
 return 1;
 else
 return 0;
}
int isempty()
{
 if(top==-1)
 return 1;
 else
 return 0;
}
```

Slip.30

Que.2

```
//read the
data from txt
file and sort
the names
using bubble
sort
#include<stdi
o.h>
#include<strin
g.h>
typedef struct
{
char
name[30];
int age;
int salary;
}
RECORD;
RECORD
emp[100];
int
readfile(RECO
RD[]);
void
writefile(REC
ORD[],int);
int
sort(RECORD
*emp,int n);
int main()
{
 int n;

n=readfile(em
p);
 sort(emp,n);

writefile(emp,
n);
}
int
readfile(RECO
RD *a)
{
 int i=0;
 FILE *fp;
 if
((fp=fopen("3
0-
2emp_bubble
.txt","r"))!=N
ULL)
 while(!
feof(fp))
 {

fscanf(fp,"%s
%d%d",a[i].na
me,&a[i].age,
&a[i].salary);
 i++;
 }
 return i-1;
}
void
writefile(REC
ORD *a,int n)
{
 int i=0;
 FILE * fp;

if((fp=fopen("
30-
```

```
2sortedemp_
bubble.txt","
w"))!=NULL)

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

fprintf(fp,"%s
\t%d\t%d\n",
a[i].name,a[i].
age,a[i].salary
);
}

 void
swap(RECOR
D *a, RECORD
*b)
 {
 RECORD
tmp=*a;
 *a=*b;
 *b=tmp;
 }

 int
sort(RECORD
* emp,int n)
 {
 int i,j,pass;

for(pass=1;pa
ss<n;pass++)
 {
 for(i=0;i<n-
pass;i++)
 {
 if (strcmp
(emp[i].name,
emp[i+1].nam
e)>0)
 {

swap(&emp[i]
,&emp[i+1]);
 } } } }
 //make text
file name as
30-
2emp_bubble
.txt save it in
vs code
//and output
is displayed in
the 30-
2sortedemp_
bubble.txt file
```