Name :- Dhruv Vimeshkumar Gandhi

ID : 17SE02CE017

1. Write a program to create memory for int, char and float variable at a run time.

```
#include <stdio.h>
int main()
{
int   *i;
char  *c;
float *f;
i=malloc(sizeof(int));
c=malloc(sizeof(char));
f=malloc(sizeof(float));
*i=100;
*c='N';
*f=123.45f;
printf("value of i= %d\n",*i);
printf("value of c= %c\n",*c);
printf("value of f= %f\n",*f);
return 0;
}
```

2. C program to read a one dimensional array, print sum of all elements along with inputted array elements using Dynamic Memory Allocation.

```
#include <stdio.h>
int main()
{
int *ptr;
int a;
int i;
int sum;
```

```
printf("Enter a of the array: ");
scanf("%d",&a);
ptr=malloc(a*sizeof(int));
for(i=0;i<a;i++)
{
printf("Enter element D:  ",i+1);
scanf("%d",(ptr+i));
}
printf("\nEntered array elements are:\n");
for(i=0;i<a;i++)
{
printf("%d\n",*(ptr+i));
}
sum=0;
for(i=0;i<a;i++)
{
sum+=*(ptr+i);
}
printf("Sum of array elements is: %d\n",sum);
return 0;
}
```

3. Develop a structure to represent planets in the solar system. Each planet has the field for the planets name, its distance from the sun in miles and the number of moon it has. Write a program to read the data for each planet and store. Also print the name of the planet that has the highest number of moons.

```
#include<stdio.h>
struct SolarSystem
{
        char name[30];
        int d;
        int m;
}planet[8];
```

```
void main()
{
int i, mm,  p;
printf("\nEnter planets records:");
for(i=0;i<8;i++)
{
     printf("\nEnter the name of the planet: ");
    scanf("%s",  planet[i].name);
    printf("\nEnter the distance of the planet from sun: ");
     scanf("%d", &planet[i].d);
   printf("\nEnter the number of moons for planet: ");
   scanf("%d", &planet[i].m);
   }
printf("\nPlanets records: \n");
printf("\nName\tDistance\tMoon");
for(i=0;i<8;i++)
{
printf("\n%s\t%d\t%d", planet[i].n,planet[i].d, planet[i].m);
}
mm = planet[0].m;
p=0;
for(i=0;i<8;i++)
{
if(planet[i].m> mm)
  {
  mm= planet[i].m;
   p=i;
   }
  }
printf("\nThe planet having highest number of moons is %s",  planet[p].n);
}
```

4. Write a c program to Add Two Complex Numbers by Passing Structure to a Function.

```
#include <stdio.h>
struct complex
{
   float real;
   float imag;
} complex;
complex add(complex a1,complex a2);
int main()
{
   complex a1, a2, temp;
printf(" 1st complex number ");
printf("Enter real and imaginary part :");
scanf("%f %f", &a1.real, &a1.imag);

printf("2nd complex number ");
printf("Enter real and imaginary part : ");
scanf("%f %f", &a2.real, &a2.imag);

temp = add(a1, a2);
printf("Sum = %.1f + %.1fi", temp.real, temp.imag);
return 0;
}
complex add(complex a1, complex a2)
{
complex temp;
temp.real = a1.real + a2.real;
temp.imag = a1.imag + a2.imag;
return(temp);
}
```

5. Write a c program to store Information using Structures with Dynamic memory allocation.

```
#include <stdio.h>
struct c
```

```
{
   int m;
   char a[10]
};
int main()
{
struct c *ptr;
   int i, r;
printf("Enter number  R ");
scanf("%d", &r);
ptr = (struct c*) malloc (r * sizeof(struct c));
for(i = 0; i < r; ++i)
   {
printf("Enter name of the S and M");
scanf("%s %d", &(ptr+i)->s, &(ptr+i)->m);
}
printf("Displaying Information:\n");
for(i = 0; i < r ; ++i)
printf("%s %d ", (ptr+i)->s, (ptr+i)->m);
return 0;
}
```

6. Write program to perform Insertion sort.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int i,j,n,k,a[30];
printf("Enter the number of elements:");
scanf("%d",&n);
printf("\nEnter the elements\n");
for(i=0;i<n;i++)
{
```

```
        scanf("%d",&a[i]);
}
for(i=1;i<=n-1;i++)
{
k=a[i];
j=i-1;
while((k<a[j])&&(j>=0))
{
a[j+1]=a[j];
j=j-1;
}
a[j+1]=k;
}
printf("\nSorted list is as follows\n");
for(i=0;i<n;i++)
{
printf("%d ",a[i]);
}
return 0;
}
```

7. Write a program to perform Selection sort.

```
#include<stdio.h>
int main()
{
        int A[5], N, Temp, i, j;
        printf("\n\n\t ENTER THE NUMBER OF TERMS...: ");
        scanf("%d",&N);
        printf("\n\t ENTER THE ELEMENTS OF THE ARRAY...:");
        for(i=1; i<=N; i++)
        {
                scanf("\n\%d", &A[i]);
        }
```

```
    for(i=1; i<=N-1; i++)
        for(j=i+1; j<=N;j++)
            if(A[i]>A[j])
            {
                Temp = A[i];
                A[i] = A[j];
                A[j] = Temp;
            }
    printf("\n THE ASCENDING ORDER LIST IS...:\n");
    for(i=1; i<=N; i++)
        printf("\n%d",A[i]);
}
```

8. Write a program to perform Bubble sort.

```
#include <stdio.h>
void bubbleSort(int arr[], int n)
{
   int i, j, temp;
   for(i = 0; i < n; i++)
   {
      for(j = 0; j < n-i-1; j++)
      {
         if( arr[j] > arr[j+1])
            {
           temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
         }
      }
   }
printf("Sorted Array: ");
   for(i = 0; i < n; i++)
   {
```

```
        printf("%d  ", arr[i]);
    }
}
void main()
{
    int arr[100], i, n, step, temp;
        printf("Enter the number of elements to be sorted: ");
        scanf("%d", &n);
   for(i = 0; i < n; i++)
    {
        printf("Enter element no. %d: ", i+1);
        scanf("%d", &arr[i]);
    }
            bubbleSort(arr, n);
}
```

9. Write a program to perform linear search.

```
#include <stdio.h>
void main()
{
    int a[5], i, n;
    printf("\nEnter FIVE elements of an array:\n");
    for (i=0; i<=5; i++)
        scanf("%d", &a[i]);
printf("\nEnter item to search: ");
scanf("%d", &n);
for (i=0; i<=5; i++)
if (n== a[i])
        {
            printf("\nItem found at location %d", i+1);
            break;
        }
        if(i > 5)
```

```
    printf("\nItem does not exist.");
}
```

10. Write a program to perform binary search.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
  int low, high, mid, size, i, e, list[100];
printf("Enter the size of the list: ");
  scanf("%d",&size);
printf("Enter %d integer values in Assending order\n", size);
for (i = 0; i < size; i++)
    scanf("%d",&list[i]);
printf("Enter value to be search: ");
  scanf("%d", &e);
while (low <= high)
   {
   if (list[middle] < sElement)
      low = high + 1;
            else if (list[mid] == sElement)
            {
            printf("Element found at index %d.\n",mid);
               break;
            }
                  else
                  high = mid - 1;
                  mid = (low + high)/2;
      if (low > high)
                  printf("Element Not found in the list.");
}
```

11. Write a program to implement stack and perform push, pop operation.

```
#include <stdio.h>
```

```
#include <stdlib.h>

void push();
void pop();
void display();


int ele,stack;

struct node
{
   int data;
   struct node *link;
}*top;



int main()
{
   pop();
    push();
    push();
   display();
}
void push()
{
   int a;
   if(top == NULL)
   {
   struct node *p;
   p = malloc(sizeof(struct node));
   p->data = 10;
   p->link = NULL;
```

```
    top=p;
    }

    else
    {

    struct node *q;
    q = malloc(sizeof(struct node));
    printf("enter value  : ");
    scanf("%d",&a);
    q->data = a;
    q->link = top;
    top=q;
    }
};



void pop()
{
    struct node *temp;
    temp=top;
    printf("%d",temp->data);
    top= top-> link;
    remove(temp);
};
void display()
{
    struct node *temp;
    temp = top;

    while(temp != NULL)
        {
```

```
        printf
        ("\n%d\n",temp->data);
        temp = temp->link;
        }
};
```

12. Write a program to perform the following operations in linear queue – Addition, Deletion and Traversing.

```
#include <stdio.h>
#include <stdlib.h>

int queue[5],front=-1,rear=-1,i,ele;
void insert();
void delete();
void display();
int main()
{
   int a;
   do{
      printf("\nHELLO");
      printf("\nENTER 1 FOR Insertion");
      printf("\nENTER 2 FOR Deletion");
      printf("\nENTER 3 FOR display");
      printf("\nENTER 4 FOR exit");
      printf("\nENTER your choice ");
      scanf("%d",&a);
      switch(a)
      {
      case 1: insert();break;
      case 2: delete(); break;
      case 3: display();break;
      case 4: break;
      }
   }while(a!=4);
```

```
        return 0;
}
void insert()
{
    if(rear==4)
    {
        printf("\nyour queue is full");
    }
   else if(front==-1)
   {
        front=0;
        rear=0;
        printf("\nplease insert the element");
        scanf("%d",&ele);
        queue[rear]=ele;
   }
    else
    {
        rear++;
        printf("\nplease insert the element");
        scanf("%d",&ele);
        queue[rear]=ele;
    }
}
void delete()
{
     if(front==-1 || front==rear+1)
    {
        printf("\nempty");
    }
    else
    {
        ele=queue[front];
```

```
        printf("\nvalue is deleted %d",ele);
        front++;
    }
}

void display()
{
    for(i=front;i<=rear;i++)
        printf(" elemnts are %d \n",queue[i]);
}
```

13. Write a program to perform the following operations in circular queue – Addition, Deletion, and Traversing.

```
#include <stdio.h>
#include <stdlib.h>
int quee[10],i,front=-1,rear=-1,ele,max=10;
void insert();
void deletion();
void display();
int main()
{
    int c;
    do
    {

    printf("enter\n1 for insert,\n2 for display,\n3 for delete  \n");
    scanf("%d",&c);
    switch (c)
     {

    case 1:
        insert();
        break;
    case 2:
```

```
        display();
        break;
    case 3:
        deletion();
        break;

    }}
    while(c!=4);

}

void insert()
{
    if (front==(rear+1)%max)
    {



    printf("the q is full");
    }
    else if(front==-1)
    {

        front= 0;
        rear=0;
        printf("enter the value");
        scanf("%d",&ele);
        quee[rear]=ele;
    }
    else
    {
         rear=(rear+1)%max;
         printf("enter the value");
         scanf("%d",&ele);
```

```
        quee[rear]=ele;

    }

}
void deletion()
{
    if(front==-1)
    {

        printf("the quee is empty");
    }
    else
    {
        ele=quee[front];
        printf("value deleted is %d",ele);
        if(front==rear)
        {
            front=-1;
            rear=-1;
        }
        else
        front=(front+1)%max;


    }
}
void display()
{
for(i=front;i<=rear;i++)
    if(front==-1)
    printf("elements are deleted");
    else
```

```
    printf("%d\n",quee[i]);
}
```

14. Write a program to perform the following operations in singly linked list – Creation, Insertion, and Deletion.

```
#include <stdio.h>
#include <stdlib.h>
void insert_first_node();
void insert_begin();
void display();
void insert_last();
void insert_between();
struct node{
        int data;
        struct node *link;
}*start;

int main()
{
    insert_first_node();
    insert_begin();
    insert_last();
    insert_between();
    display();
    return 0;
}
void insert_first_node(){
    struct node *p;
    p= malloc(sizeof(struct node));
    p->data=10;
    p->link=NULL;
    start=p;
}
void insert_begin(){
```

```
struct node *p;
   p= malloc(sizeof(struct node));
   p->data=20;
   p->link=start;
   start=p;
}

void insert_last()
{
struct node *q;
q=start;
while(q->link!=NULL)
{
   q=q->link;
}
struct node *s;
s=malloc(sizeof(struct node));
s->data=30;
q->link=s;
s->link=NULL;
}
void insert_between()
{
   int pos;
   struct node *r,*s;
   r=start;
   printf("enter a position name:");
   scanf("%d",&pos);
   for(int i=1;i<pos;i++)
   {
      r=r->link;
   }
```

```
        s=malloc(sizeof(struct node));
        s->data=50;
        s->link=r->link;
        r->link=s;
}

void display()
{
struct node *temp;
temp=start;
while(temp !=NULL)
{
printf("\n%d\n",temp->data);
temp = temp->link;
}
}
```

15. Write a program to perform the following operations in doubly linked list – Creation, Insertion, and Deletion.

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int data;
   struct node *next;
   struct node *previous;
} *p,*start;

void insert(){
struct node *p;
p = malloc(sizeof (struct node));
p->data=10;
p->next=NULL;
```

```
p->previous=NULL;
start=p;
}

void insert_beg(){
struct node *p1;
p1 = malloc(sizeof (struct node));
p1->data=5;
p1->previous=NULL;
p1->next=start;
start->previous=p1;
start=p1;

}

void display(){
struct node *temp;
temp=start;
while(temp!=NULL)
   {
   printf("\n%d",temp->data);
   temp=temp->next;
   }
}

void insert_last(){

struct node *temp;
temp=start;
while(temp->next!=NULL)
{
   temp=temp->next;
}
```

```
struct node *p2;
p2 = malloc(sizeof (struct node));
p2->data=15;
temp->next=p2;
p2->previous=temp;
p2->next=NULL;

}

void insertbetween(){

   struct node *temp;
   temp=start;
   while(temp->data!=10)
   {
      temp=temp->next;
   }
   struct node *between;
      between = malloc(sizeof (struct node));
      between->data=500;
      between->next=temp->next;
      temp->next=between;
      between->previous=temp;
      temp->next->previous=between;



}


int main()
{
```

```
    insert();
    insert_beg();
    insert_last();
    insertbetween();
    display();
    return 0;
}
```

16. Write programs to implement linked stack.

```
#include <stdio.h>
struct node{
    int data;
    struct node *link;
}*top;
void push(){

    if(top==NULL)
    {
        struct node *p;
        p=malloc(sizeof (struct node));
        p->data=10;
        p->link=NULL;
        top=p;
    }
    else{
        int d;
        struct node *q;
        q=malloc(sizeof (struct node));
        printf("enter element");
        scanf("%d",&d);
        q->data=d;
        q->link=top;
        top=q;
```

```
    }


}
void display(){

    struct node *temp;
    temp=top;
    while(temp!=NULL){
        printf("\n%d",temp->data);

       temp=temp->link;

    }
}

void deletion(){
    struct node *temp;
    temp=top;
    printf("deleted element is:%d",temp->data);
    top=top->link;
    remove(temp);




}
int main()
{
    int n,i;
    printf("enter value of n");
    scanf("%d",&n);
    for(i=0;i<=n;i++){
        push();
```

```
    }
display();
deletion();
deletion();
display();
    return 0;
}
```

17. Write programs to implement linked queue.

```
#include <stdio.h>
struct node
{
    int data;
    struct node *link;
}*rear,*front;

void insert(){
    if(rear==NULL)
    {
        struct node *p;
        p=malloc(sizeof (struct node));
        p->data=10;
        p->link=NULL;
        front=p;
        rear=p;
    }
    else
    {
        int d;
        struct node *q;
        q=malloc(sizeof (struct node));
```

```
        printf("enter element");
        scanf("%d",&d);
        q->data=d;
        q->link=NULL;
        rear->link=q;
        rear=q;
    }
}

void display(){
        struct node *temp;
        temp=front;
        while(temp!=NULL){
        printf("\n%d",temp->data);
        temp=temp->link;
        }

}

void deletion(){
    struct node *temp;
    temp=front;
    printf("deleted element is:%d",temp->data);
    front=front->link;
    remove(temp);
}


int main()
{
    int i;
    for(i=0;i<5;i++){
    insert();
```

```
    }
    display();
    deletion();
    display();
    return 0;
}
```

18.Write a program to create a binary search tree and perform – Insertion, Deletion, and Traversal.

```
#include <iostream>

using namespace std;
struct node{
int data;
struct node *l,*r;
};
void create(struct node **bt,int num);

void inorder(struct node *);
void postorder(struct node *);
void preorder(struct node *);
int main()
{
    struct node *sr;
    sr=NULL;

create(&sr,2);
create(&sr,1);
create(&sr,3);
  printf("inorder:\n");
inorder(sr);
printf("postorder:\n");
postorder(sr);
printf("preorder:\n");
```

```
preorder(sr);
    return 0;
}
void create(struct node **bt,int num)
{


    if(*bt==NULL)
    {
        *bt=(struct node *)malloc(sizeof(struct node));
        (*bt)->data=num;
        (*bt)->l=NULL;
        (*bt)->r=NULL;
    }
    else
    {
        if(num<(*bt)->data)
        create((&(*bt)->l),num);
        else
        create((&(*bt)->r),num);

    }


}
void inorder(struct node *bt)
{

    if(bt!=NULL)

    {
```

```
        inorder(bt->l);
        printf("%d\n",bt->data);
        inorder(bt->r);

    }
}
void postorder(struct node *bt)
{

    if(bt !=NULL)
    {
        postorder(bt->l);
        postorder(bt->r);
        printf("%d\n",bt->data);
    }
}
void preorder(struct node *bt)
{

    if(bt !=NULL)
    {
        printf("%d\n",bt->data);
        postorder(bt->l);
        postorder(bt->r);

    }
}
```

19. Write a program to perform the following operations in circular linked list – Creation, Insertion and Deletion.

```
#include <stdio.h>
#include <stdlib.h>
```

```
void insert_first_node();
void insert_begin();
void insert_last();
void insert_between();
void display();

struct node{
        int data;
        struct node *next;
}*head;
int main()
{

     insert_first_node();
     insert_begin();
insert_last();
    insert_between();

     display();
    return 0;
}
void insert_first_node()
{
    struct node *p;
    p=malloc(sizeof(struct node));
     p->data=10;
       //if(head == NULL)
    //{

       p -> next = p;
       head = p;
    //}
```

```
}

void insert_begin()
{
   struct node *p;
   p=malloc(sizeof(struct node));
   p->data=20;
   p->next =head;
   head->next=p;
   head=p;
}

void insert_last()
{
   struct node *q,*temp;
temp=head;
      while(temp->next!=head)
      {
         temp=temp->next;

      }
      q=malloc(sizeof(struct node));
      q->data=100;

      q->next=head;
      temp->next=q;


}

void insert_between()
{
   int pos;
```

```
    struct node *r,*s;
    r=head;
    printf("enter a position name:");
    scanf("%d",&pos);
    for(int i=1;i<pos;i++)
    {
        r=r->next;
    }

        s=malloc(sizeof(struct node));
        s->data=50;
        s->next=r->next;
        r->next=s;
}


void display()
{
struct node *temp;
temp=head;
    do
        {
    printf("\n%d\n",temp->data);
    temp=temp->next;
        }
    while(temp!=head);
}
```