Assignment: 8

Name: Ujjwal Kumar Jha

Registration No. 20194196

Group: C_2

1. **WAP to implement Sparse Matrix using**
   a. Triplet representation.

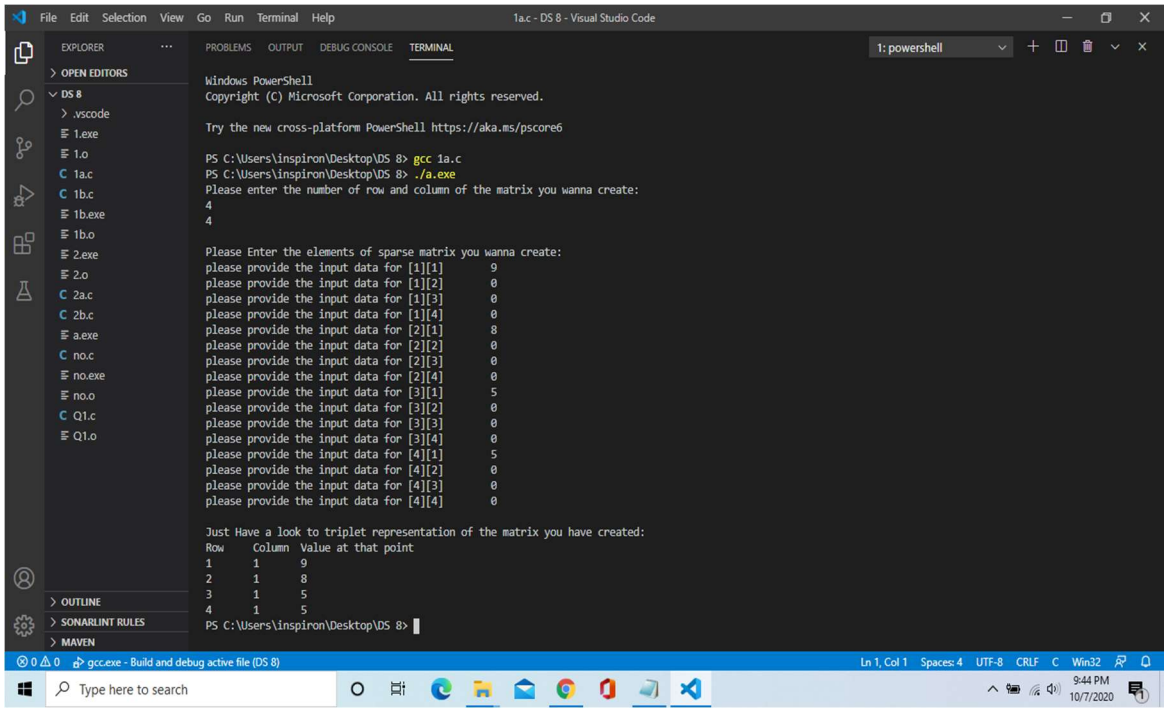
```
#include <stdio.h>
#include <stdlib.h>
int **sp,**trip,m,n,count;
int main(){
printf("Please enter the number of row and column of the matrix you wanna create:
\n");
scanf("%d%d",&m,&n);




int i,j;
sp=(int **)malloc(m*sizeof(int *));
for(i=0;i<m;i++)
    *(sp+i)=(int *)malloc(sizeof(int));
printf("\nPlease Enter the elements of sparse matrix you wanna create:\n");
for(i=0;i<m;i++)
    for(j=0;j<n;j++){
        printf("please provide the input data for [%d][%d]\t",i+1,j+1);
    scanf("%d",&sp[i][j]);}

int k=0;
count=0;
for(i=0;i<m;i++)
    for(j=0;j<n;j++)
    if(sp[i][j]!=0)
    count++;
trip=(int **)malloc((count+1)*sizeof(int));
for(i=0;i<(count+1);i++)
```

```
*(trip+i)=(int *)malloc(3*sizeof(int));
trip[0][0]=m;
trip[0][1]=n;
trip[0][2]=count;
for(i=0;i<m;i++)
for(j=0;j<n;j++)
if(sp[i][j]!=0){
    trip[k][0]=i+1;
    trip[k][1]=j+1;
    trip[k][2]=sp[i][j];k++;
}
printf("\nJust Have a look to triplet representation of the matrix you have creat
ed:\n");
printf("Row\tColumn\tValue at that point\n");
for(i=0;i<(count);i++){
    for(j=0;j<3;j++)
    printf("%d\t",trip[i][j]);
printf("\n");
}
}
```



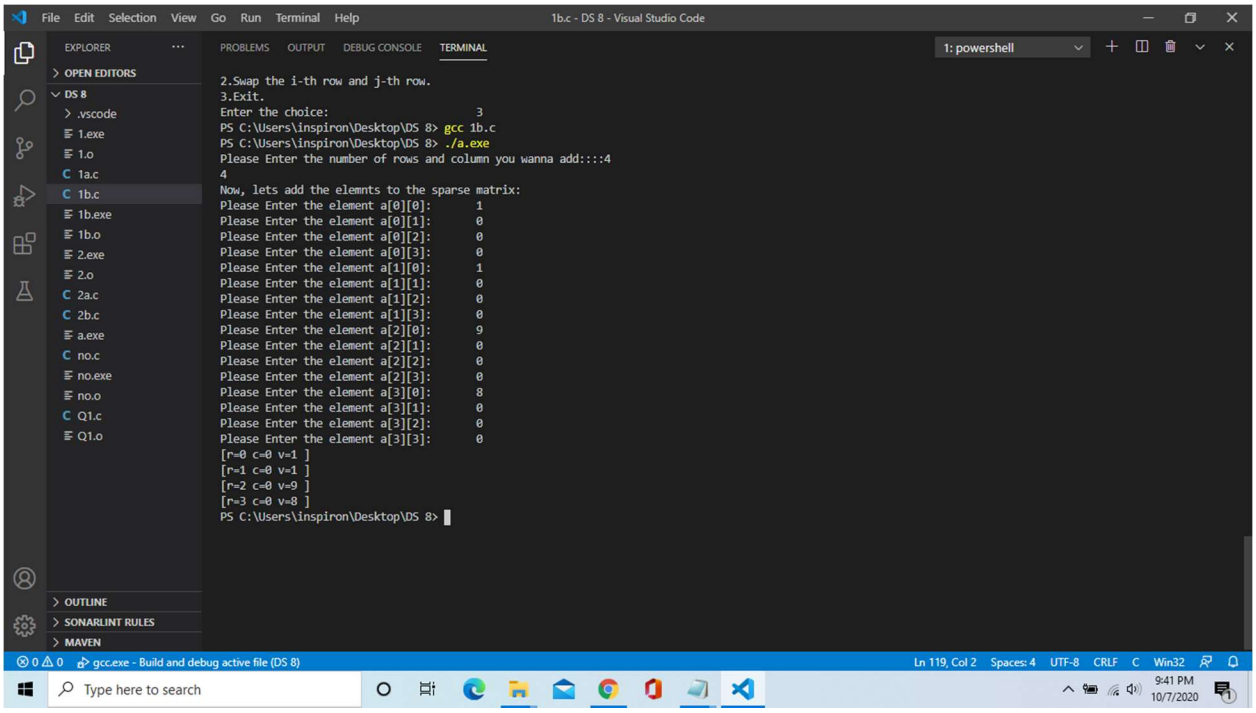
b. Multi-linked Representation:

```
#include <stdio.h>
#include <malloc.h>
#define node struct node
node
{
    int row,col,value;
    node *nextrow,*nextcol;
};
node *head=NULL;
void create_MLlist(int m,int n)
{
    node *newnode,*ptr;
    int i=0;
    if(head==NULL)
    {
        head=(node *)malloc(sizeof(node));
        head->row=-1;
        head->col=-1;
    }
    ptr=head;
    while(n--)
    {
        newnode=(node *)malloc(sizeof(node));
        newnode->col=i++;
        newnode->row=-1;
        newnode->nextrow=newnode;
        ptr->nextcol=newnode;
        newnode->nextcol=head;
        ptr=newnode;
    }
    i=0;
    ptr=head;
    while(m--)
    {
        newnode=(node *)malloc(sizeof(node));
        newnode->row=i++;
        newnode->col=-1;
        newnode->nextcol=newnode;
        ptr->nextrow=newnode;
        newnode->nextrow=head;
        ptr=newnode;
    }
}
```

```
node *above(node *head,int r,int c)
{
    node *p=head,*q;
    while(p->col!=c)
    p=p->nextcol;
    do
    {
        q=p;
        p=p->nextrow;
    }while(p->row<r && p->row!=-1);
    return q;
}
node *left(node *head,int r,int c)
{
    node *p=head,*q;
    while(p->row!=r)
    p=p->nextrow;
    do{
        q=p;
        p=p->nextcol;
    }while(p->col<c && p->col!=-1);
    return q;
}
void insertion(int r,int c,int v)
{
    node *a,*l,*newnode;
    a=above(head,r,c);
    l=left(head,r,c);
    newnode=(node *)malloc(sizeof(node));
    newnode->row=r;
    newnode->col=c;
    newnode->value=v;
    newnode->nextrow=a->nextrow;
    newnode->nextcol=l->nextcol;
    a->nextrow=newnode;
    l->nextcol=newnode;
}
void display()
{
    node *ptr=head->nextrow,*q=ptr->nextcol;
    while(ptr->row!=-1)
    {
        while(q->col!=-1)
        {
            printf("[r=%d c=%d v=%d ]  ",q->row,q->col,q->value);
```

```
            q=q->nextcol;
        }
        printf("\n");
        ptr=ptr->nextrow;
        q=ptr->nextcol;
    }
}
int main()
{
    int r,c,opt;
    printf("Please Enter the number of rows and column you wanna add::::");
    scanf("%d %d",&r,&c);
    create_MLlist(r,c);
    printf("Now, lets add the elemnts to the sparse matrix:\n");
    int i,j,x,y;
    int a[r][c];
    for(i=0;i<r;i++)
    {

        for(j=0;j<c;j++)
        {
            printf("Please Enter the element a[%d][%d]:\t",i,j);
            scanf("%d",&a[i][j]);
            if(a[i][j]!=0)
            {
                insertion(i,j,a[i][j]);
            }
        }
    }
    display();
}
```



2. WAP to implement the following for both the representations.

a. Delete the i-th row from the Sparse Matrix.

b. Swap the i-th row and j-th row of the Sparse Matrix.

i. Triplet Representation:

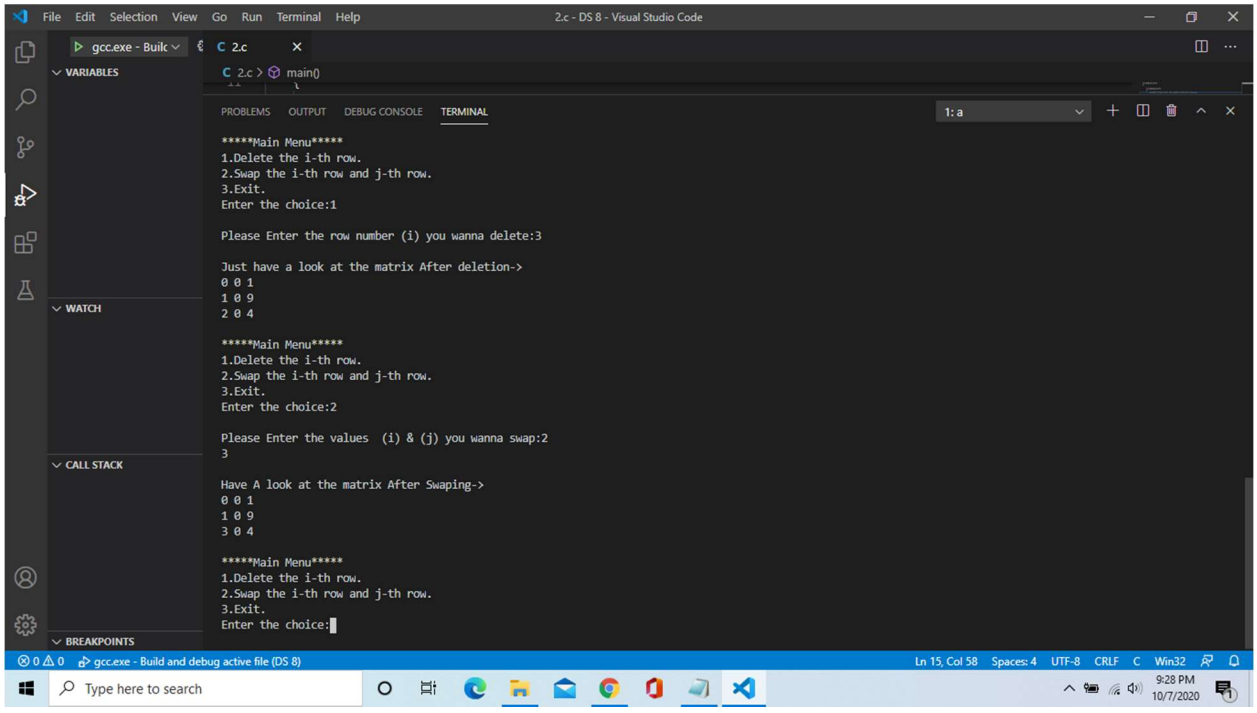
```c
#include <stdio.h>
int main()
{
    int r,c,opt;
    printf("Please Enter the number of rows and column you wanna add::");
    scanf("%d %d",&r,&c);
    printf("Now, lets add the elemnts to the sparse matrix:\n");
    int i,j,n=0,x=0,y,I,J;
    int a[r][c];
    for(i=0;i<r;i++)
    {

        for(j=0;j<c;j++)
```

```
        {
            printf("Please Enter the element a[%d][%d]:\t",i,j);
            scanf("%d",&a[i][j]);
            if(a[i][j]!=0)
            {
                n++;
            }
        }
    }
    int t[n][3];
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            if(a[i][j]!=0)
            {
                t[x][0]=i;
                t[x][1]=j;
                t[x][2]=a[i][j];
                x++;
            }
        }
    }
    do
    {
        printf("\n*****Main Menu*****");
        printf("\n1.Delete the i-th row.");
        printf("\n2.Swap the i-th row and j-th row.");
        printf("\n3.Exit.");
        printf("\nEnter the choice:");
        scanf("%d",&opt);
        switch(opt)
        {
            case 1:
                printf("\nPlease Enter the row number (i) you wanna delete:");
                scanf("%d",&x);
                for(i=0;i<n;i++)
                {
                    if(x==t[i][0])
                    {
                        for(j=i+1;j<n;j++)
                        {
                            t[j-1][0]=t[j][0];
                            t[j-1][1]=t[j][1];
                            t[j-1][2]=t[j][2];
```

```
                    }
                    n=n-1;
                    i=i-1;
                }
            }
            printf("\nJust have a look at the matrix After deletion->\n");
            for(i=0;i<n;i++)
            {
                for(j=0;j<3;j++)
                printf("%d ",t[i][j]);
                printf("\n");
            }
            break;
        case 2:
            printf("\nPlease Enter the values  (i) & (j) you wanna swap:");
            scanf("%d %d",&x,&y);
            I=x<y?x:y;
            J=x>y?x:y;
            for(i=0;i<n;i++)
            {
                    if(t[i][0]==I)
                    {
                        t[i][0]=J;
                    }
                    else if(t[i][0]>I)
                    break;
            }
            for(j=i;j<n;j++)
            {
                if(t[j][0]==J)
                {
                    t[j][0]=I;
                }
            }
            //Sorting
            for(i=0;i<n-1;i++)
            {
                for(j=0;j<n-i-1;j++)
                {
                    if (t[j][0] > t[j+1][0])
                    {
                        int tr=t[j+1][0];
                        int tc=t[j+1][1];
                        int tv=t[j+1][2];
                        t[j+1][0]=t[j][0];
```

```
                        t[j+1][1]=t[j][1];
                        t[j+1][2]=t[j][2];
                        t[j][0]=tr;
                        t[j][1]=tc;
                        t[j][2]=tv;
                    }
                }
            }
            printf("\nHave A look at the matrix After Swaping->\n");
            for(i=0;i<n;i++)
            {
                for(j=0;j<3;j++)
                printf("%d ",t[i][j]);
                printf("\n");
            }
            break;
        }
    }while(opt!=3);
}
```



ii. Multi linked representation:

```
#include <stdio.h>
#include <malloc.h>
#define node struct node
node
{
    int row,col,value;
    node *nextrow,*nextcol;
};
node *head=NULL;
void create_MLlist(int m,int n)
{
    node *newnode,*ptr;
    int i=0;
    if(head==NULL)
    {
        head=(node *)malloc(sizeof(node));
        head->row=-1;
        head->col=-1;
    }
    ptr=head;
    while(n--)
    {
        newnode=(node *)malloc(sizeof(node));
        newnode->col=i++;
        newnode->row=-1;
        newnode->nextrow=newnode;
        ptr->nextcol=newnode;
        newnode->nextcol=head;
        ptr=newnode;
    }
    i=0;
    ptr=head;
    while(m--)
    {
        newnode=(node *)malloc(sizeof(node));
        newnode->row=i++;
        newnode->col=-1;
        newnode->nextcol=newnode;
        ptr->nextrow=newnode;
        newnode->nextrow=head;
        ptr=newnode;
```

```
    }
}
node *above(node *head,int r,int c)
{
    node *p=head,*q;
    while(p->col!=c)
    p=p->nextcol;
    do
    {
        q=p;
        p=p->nextrow;
    }while(p->row<r && p->row!=-1);
    return q;
}
node *left(node *head,int r,int c)
{
    node *p=head,*q;
    while(p->row!=r)
    p=p->nextrow;
    do{
        q=p;
        p=p->nextcol;
    }while(p->col<c && p->col!=-1);
    return q;
}
void insertion(int r,int c,int v)
{
    node *a,*l,*newnode;
    a=above(head,r,c);
    l=left(head,r,c);
    newnode=(node *)malloc(sizeof(node));
    newnode->row=r;
    newnode->col=c;
    newnode->value=v;
    newnode->nextrow=a->nextrow;
    newnode->nextcol=l->nextcol;
    a->nextrow=newnode;
    l->nextcol=newnode;
}
void deletion(int r,int c)
{
    node *p,*a,*l;
    a=above(head,r,c);
    l=left(head,r,c);
    p=a->nextrow;
```

```
    a->nextrow=p->nextrow;
    l->nextcol=p->nextcol;
    free(p);
}
void delete_ith_row(int r)
{
    node *p=head,*q;
    while(p->row!=r)
    p=p->nextrow;
    q=p->nextcol;
    while(p!=q)
    {
        deletion(q->row,q->col);
        q=p->nextcol;
    }
}

void swap_ithrow_with_jthrow(int ri,int rj)
{
    node *p=head->nextrow,*pi,*pj;
    while(p->row!=-1)
    {
       if(p->row==ri)
       pi=p;
       if(p->row==rj)
       pj=p;
       p=p->nextrow;
    }
    pi=pi->nextcol;
    pj=pj->nextcol;
    while(pi->col!=-1 && pj->col!=-1)
    {
        if(pi->col==pj->col)
        {
            int t=pi->value;
            pi->value=pj->value;
            pj->value=t;
            pi=pi->nextcol;
            pj=pj->nextcol;
        }
        else if(pi->col<pj->col)
        {
            insertion(pj->row,pi->col,pi->value);
            int r=pi->row;
            int c=pi->col;
```

```
            pi=pi->nextcol;
            deletion(r,c);
        }
        else
        {
            insertion(pi->row,pj->col,pj->value);
            int r=pj->row;
            int c=pj->col;
            pj=pj->nextcol;
            deletion(r,c);
        }
    }
    while(pi->col!=-1)
    {
        insertion(pj->row,pi->col,pi->value);
        int r=pi->row;
        int c=pi->col;
        pi=pi->nextcol;
        deletion(r,c);
    }
    while(pj->col!=-1)
    {
        insertion(pi->row,pj->col,pj->value);
        int r=pj->row;
        int c=pj->col;
        pj=pj->nextcol;
        deletion(r,c);
    }
}


void display()
{
    node *ptr=head->nextrow,*q=ptr->nextcol;
    while(ptr->row!=-1)
    {
        while(q->col!=-1)
        {
            printf("[r=%d c=%d v=%d ]  ",q->row,q->col,q->value);
            q=q->nextcol;
        }
        printf("\n");
        ptr=ptr->nextrow;
        q=ptr->nextcol;
    }
```

```
}
int main()
{
    int r,c,opt;
    printf("Please Enter the number of rows and column you wanna add::::");
    scanf("%d %d",&r,&c);
    create_MLlist(r,c);
    printf("Now, lets add the elemnts to the sparse matrix:\n");
    int i,j,x,y;
    int a[r][c];
    for(i=0;i<r;i++)
    {

        for(j=0;j<c;j++)
        {
            printf("Please Enter the element a[%d][%d]:\t",i,j);
            scanf("%d",&a[i][j]);
            if(a[i][j]!=0)
            {
                insertion(i,j,a[i][j]);
            }
        }
    }
    do
    {
        printf("\n*****Main Menu*****");
        printf("\n1.Delete the i-th row.");
        printf("\n2.Swap the i-th row and j-th row.");
        printf("\n3.Exit.");
        printf("\nEnter the choice:");
        scanf("%d",&opt);
        switch(opt)
        {
            case 1:
                printf("\nPlease Enter the row number (i) you wanna delete::");
                scanf("%d",&x);
                delete_ith_row(x);
                printf("\nJust have a look at the matrix After deletion->\n");
                display();
                break;
            case 2:
                printf("\nPlease Enter the values  (i) & (j) you wanna swap:");
                scanf("%d %d",&x,&y);
                swap_ithrow_with_jthrow(x,y);
                printf("\nHave A look at the matrix After Swaping->\n");
```

```
                display();
                break;
        }
    }while(opt!=3);
}
```