NAME- SHUBHAM GARG

ENROLL . NO- 9919103057

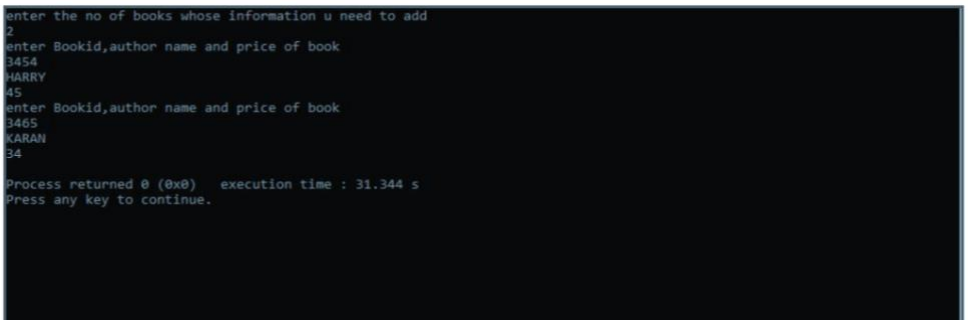
ASSIGNMENT-I

QUESTION 1

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
 FILE *fp;
int i,n,arr[100],j,temp,ind;
struct book
{
int bookid;
char authorname[20];
int price;
};
printf("enter the No. of Books\n");
scanf("%d",&n);
struct book b[n];
fp=fopen("file.c","w");
if(fp==NULL)
{
printf("cant open file\n");
exit(0);
}
for(i=0;i<n;i++)
{
printf("Enter Bookid,author name and price of book\n");
scanf("%d%s%d",&b[i].bookid,b[i].authorname,&b[i].price);
fprintf(fp,"%d%s%d",b[i].bookid,b[i].authorname,b[i].price);
fflush(stdin);
}
```

```
fclose(fp);
fp=fopen("file.C","r");
 if(fp==NULL)
 {
 printf("cnat open file\n");
 exit(0);
 }
 for(i=0;i<n;i++)
 {
 arr[i]=fscanf(fp,"%d",&b[i].price);
 exit(0);
 }
 for(i=0;i<n-1;i++)
 {
 for(j=0;j<n-i-1;j++)
 {
 if(arr[j]>arr[j+1])
 {
 temp=arr[j+1];
 arr[j+1]=arr[j];
 arr[j]=temp;
 }
 }
 }
 for(i=0;i<n;i++)
 {
 if(arr[1]==fscanf(fp,"%d",&b[i].price));
 {
 printf("displaying the details of book having second min price\n");
 fscanf(fp,"%d%s%d",&b[i].bookid,&b[i].authorname,&b[i].price);
 printf("bookid %d \n authorname %s \n price %d
 \n",b[i].bookid,b[i].authorname,b[i].price);
 ind=i;
 }
 }
 fclose(fp);
 fp=fopen("file.c","a+");
 if(fp==NULL)
 {
```

```
printf("cnt open file\n");
exit(0);
}
fscanf(fp,"%d%s%d",&b[ind].bookid,&b[ind].authorname,
&b[ind].price);
fprintf(fp,"bookid %d \n authorname %s \n price %d
\n",b[ind].bookid,b[ind].authorname,b[ind].price);
fclose(fp);
}
```



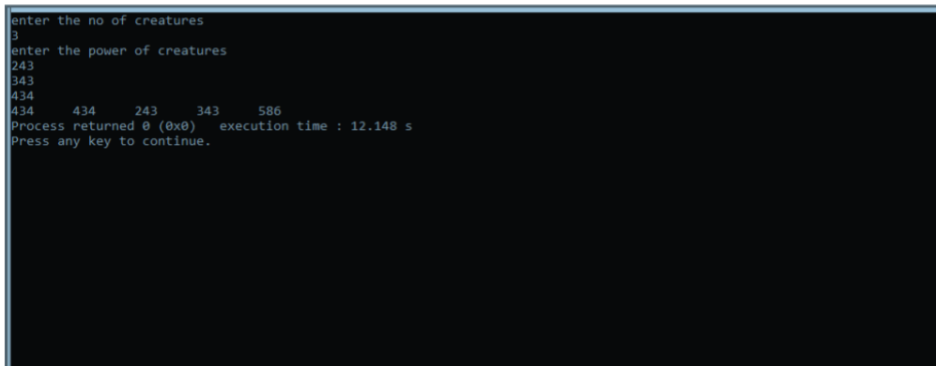
QUESTION-2

```
#include<stdio.h>
int main()
{
int i,j=0,k=0,l,n,temp,vampire[100],zombie[100],sumv=0,sumz=0,power;
printf("enter the no of creatures\n");
scanf("%d",&n);
printf("enter the power of creatures\n");
for(i=1;i<=n;i++)
{
scanf("%d",&power);
if(power%2==0)
{
zombie[j]=power;
j++;
}
else
{
```

```
vampire[k]=power;
k++;
}
}
for(i=0;i<j;i++)
{
for(l=0;l<j-i-1;l++)
{
if(zombie[l]>zombie[l+1])
{
temp=zombie[l+1];
zombie[l+1]=zombie[l];
zombie[l]=temp;
}
}
}
for(i=0;i<k;i++)
{
for(l=0;l<k-i-1;l++)
{
if(vampire[l]>vampire[l+1])
{
temp=vampire[l+1];
vampire[l+1]=vampire[l];
vampire[l]=temp;
}
}
}
for(i=0;i<j;i++)
{
sumz=sumz+zombie[i];
printf("%d\t",zombie[i]);
}
printf("%d\t",sumz);
for(i=0;i<k;i++)
{
sumv=sumv+vampire[i];
printf("%d\t",vampire[i]);
```

```
}
printf("%d\t",sumv);
}
```

OUTPUT



QUESTION-3

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
int coeff;
int pow;
struct Node *next;
};
void create_node(int x, int y, struct Node **temp)
{
struct Node *r, *z;
z = *temp;
if(z == NULL)
{
r =(struct Node*)malloc(sizeof(struct Node));
r->coeff = x;
r->pow = y;
*temp = r;
r->next = (struct Node*)malloc(sizeof(struct Node));
r = r->next;
r->next = NULL;
}
else
```

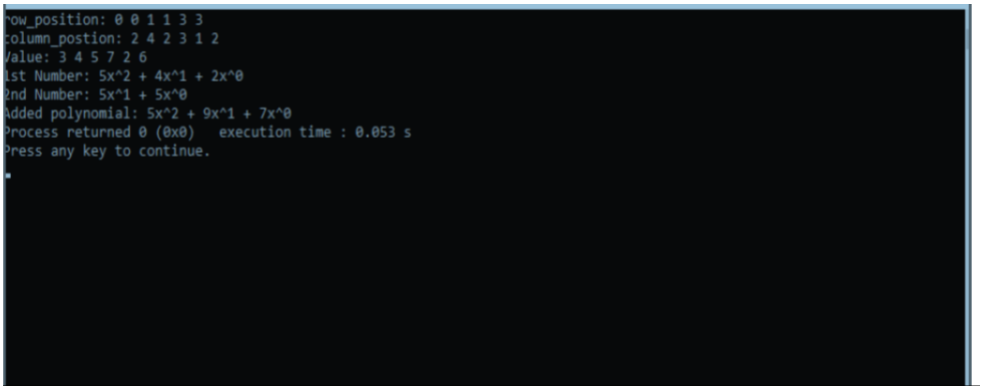
```
{
r->coeff = x;
r->pow = y;
r->next = (struct Node*)malloc(sizeof(struct Node));
r = r->next;
r->next = NULL;
}
}
void polyadd(struct Node *poly1, struct Node *poly2, struct Node *poly)
{
while(poly1->next && poly2->next)
{
if(poly1->pow > poly2->pow)
{
poly->pow = poly1->pow;
poly->coeff = poly1->coeff;
poly1 = poly1->next;
}
else if(poly1->pow < poly2->pow)
{
poly->pow = poly2->pow;
poly->coeff = poly2->coeff;
poly2 = poly2->next;
}
else
{
poly->pow = poly1->pow;
poly->coeff = poly1->coeff+poly2->coeff;
poly1 = poly1->next;
poly2 = poly2->next;
}
poly->next = (struct Node *)malloc(sizeof(struct Node));
poly = poly->next;
poly->next = NULL;
}
while(poly1->next || poly2->next)
{
if(poly1->next)
```

```
{
poly->pow = poly1->pow;
poly->coeff = poly1->coeff;
poly1 = poly1->next;
}
if(poly2->next)
{
poly->pow = poly2->pow;
poly->coeff = poly2->coeff;
poly2 = poly2->next;
}
poly->next = (struct Node *)malloc(sizeof(struct Node));
poly = poly->next;
poly->next = NULL;
}
}

void show(struct Node *node)
{
while(node->next != NULL)
{
printf("%dx^%d", node->coeff, node->pow);
node = node->next;
if(node->next != NULL)
printf(" + ");
}
}

int main()
{
struct Node *poly1 = NULL, *poly2 = NULL, *poly = NULL;
create_node(5,2,&poly1);
create_node(4,1,&poly1);
create_node(2,0,&poly1);
create_node(5,1,&poly2);
create_node(5,0,&poly2);
printf("1st Number: ");
show(poly1);
```

```
printf("\n2nd Number: ");
show(poly2);
poly = (struct Node *)malloc(sizeof(struct Node));
polyadd(poly1, poly2, poly);
printf("\nAdded polynomial: ");
show(poly);
return 0;
}
```




```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
        int value;
        int rowpos;
        int colpos;
        struct Node *next;
};

void createnode(struct Node** start, int val,
                                        int rowindex, int columnindex )
{
        struct Node *temp, *r;
        temp = *start;
        if (temp == NULL)
        {

                temp = (struct Node *) malloc (sizeof(struct Node));
                temp->value =val;
                temp->rowpos= rowindex;
```

```
            temp->colpos= columnindex;
            temp->next = NULL;
            *start = temp;

        }
        else
        {
            while (temp->next != NULL)
                temp = temp->next;

            r = (struct Node *) malloc (sizeof(struct Node));
            r->value = val;
            r->rowpos= rowindex;
            r->colpos= columnindex;
            r->next = NULL;
            temp->next = r;

        }
}

void PrintList(struct Node* start)
{
        struct Node *temp, *r, *s;
        temp = r = s = start;

        printf("row_position: ");
        while(temp != NULL)
        {

            printf("%d ", temp->rowpos);
            temp = temp->next;
        }
        printf("\n");

        printf("column_postion: ");
        while(r != NULL)
        {
            printf("%d ", r->colpos);
```

```
            r = r->next;
        }
        printf("\n");
        printf("Value: ");
        while(s != NULL)
        {
            printf("%d ", s->value);
            s = s->next;
        }
        printf("\n");
}

int main()
{  int row,col;
   printf("Enter no of rows and columns");
   scanf("%d %d",&row,&col);
   int sparseMatric[row][col];
   for(int i=0;i<row;i++)
      for(int j=0;j<col;j++)
       scanf("%d",&sparseMatric[i][j]);
        struct Node* start = NULL;

        for (int i = 0; i < row; i++)
            for (int j = 0; j < col; j++)

                if (sparseMatric[i][j] != 0)
                    createnode(&start, sparseMatric[i][j], i, j);

        PrintList(start);

        return 0;
}
```