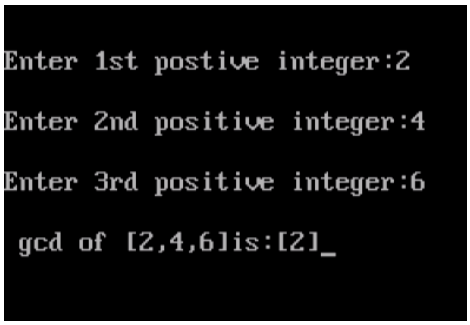
[1] Write c program to find the GCD of two number using recursion.

```
#include<stdio.h>
#include<stdlib.h>
/*recursive function*/
int gcd(int i,int j)
{
if(j>i)
return gcd(j,i);
if(j==0)
return i;
else
return gcd(j,i%j);
}
int main()
{
int num1,num2,num3,gcd1,gcd2;
printf("\nEnter 1st positive integer::");
scanf("%d",&num1);
printf("\nEnter 2nd positive integer::");
scanf("%d",&num2);
printf("\nEnter 3rd positive integer::");
scanf("%d",&num3);
if(num1==0&&num2==0&&num3==0)
{
printf("\ninvalid number");
exit(0);
}
gcd2=gcd(num3, gcd(num1, num2));
printf("\n gcd of [%d,%d,%d]is:[%d]/n",num1,num2,num3,gcd2);
```

```
return 0;
}
```

OUTPUT:



[2] Write a program to implement linear search in c.

```
#include<stdio.h>
#include<conio.h>
int linscr(int a[],int n,int x)
{
 int i;
 for(i=0;i<n;i++)
  if(a[i]==x)
  return i+1;
  return -1;
}
void main()
{
 int a[10],n,i,pos,x;
 clrscr();
 printf("enter the array size:");
 scanf("%d",&n);
 printf("enter the array element:");
  for(i=0;i<n;i++)
  scanf("%d",&a[i]);
 printf("enter the array search element:");
 scanf("%d",&x);
   pos=linscr(a,n,x);
   if(pos==-1)
   printf("search element is unsuccessfull");
   else
   printf("search successful element found at position: %d",pos);
   getch();
}
```

OUTPUT:

```
enter the array size:5
enter the array element:1 2 3 4 5
enter the array search element:3
search successfull element found at position: 3_
```

[3] Write a program to implement Binary search in C.

```
#include<stdio.h>
int binsrch(int a[],int low,int high,int x);
void main()
{
int a[10],n,i,pos,x;
printf("\n enter array size:");
scanf("%d",&n);
printf("\n enter array elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n enter search element:");
scanf("%d",&x);
pos=binsrch(a,0,n-1,x);
if(pos==-1)
printf("\n search unsuccessful");
else
printf("\n search is successful., elements found at index position: %d",pos);
}
int binsrch(int a[],int low,int high,int x)
{
int mid;
while(low<=high)
{
mid=(low+high)/2;
if(x<a[mid])
high=mid-1;
else
if(x==a[mid])
```

```
return mid+1;
else
low=mid+1;
}
return-1;
}
```

OUTPUT:

```
enter array size:5
enter array elements:4 5 2 3 1
enter search element:2
search is successful., elements found at index position: 3
```

[4] Write a program to implement bubble sort in C.

```
#include<stdio.h>
#include<conio.h>
void bubble(int a[],int n);
void main()
{
int a[10],n,i;
clrscr();
printf("\n enter array size:");
scanf("%d",&n);
printf("\n enter array elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
bubble(a,n);
printf("\n sorted list:");
for(i=0;i<n;i++)
printf("\t%d",a[i]);
getch();
}
void bubble(int a[],int n)
{
int i,j,temp;
for(i=0;i<n-1;i++)
for(j=0;j<n-i-1;j++)
{
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
```

```
a[j+1]=temp;
getch();
}
}
}
```

OUTPUT:

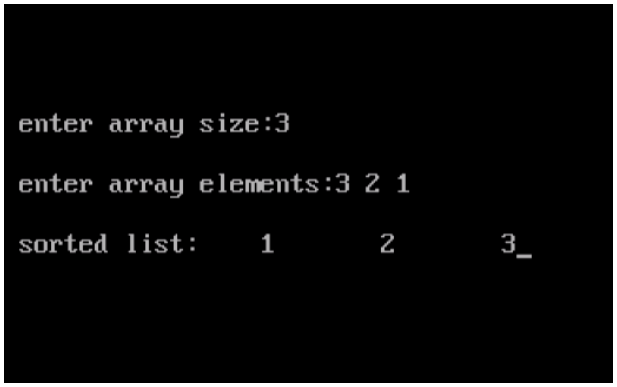
```
enter array size:5
enter array elements:4 5 3 1 2
sorted list:   1      2      3      4      5
```

[5] Write a program to implement selection sort in C.

```
#include<stdio.h>
#include<conio.h>
void selsort (int a[],int n);
void main()
{
int a[10],n,i;
clrscr();
printf("\n enter array size:");
scanf("%d",&n);
printf("\n enter array elements:");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
selsort(a,n);
printf("\n sorted list:");
for(i=0;i<n;i++)
printf("\t %d",a[i]);
}
void selsort(int a[],int n)
{
int i,j,min,pos;
for(i=0;i<n-1;i++)
{
min=a[i];
pos=i;
for(j=i+1;j<n;j++)
{
if(a[j]>min)
{
```

```
min=a[j];
pos=j;
}
}
a[pos]=a[j];
a[i]=min;
getch();
}
}
```

OUTPUT:

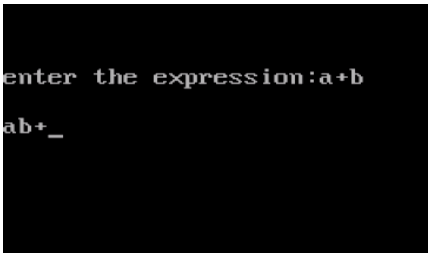


[6] Write a program to convert an infix expression to postfix expression.

```
#include<stdio.h>
#include<conio.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==')')
{
while((x=pop())!='(')
printf("%c",x);
}
else
{
while(priority(stack[top])>=priority(*e))
printf("%c",pop());
push(*e);
}
e++;
}
while(top!=-1)
{
printf("%c",pop());
}
return 0;
}
```

OUTPUT:



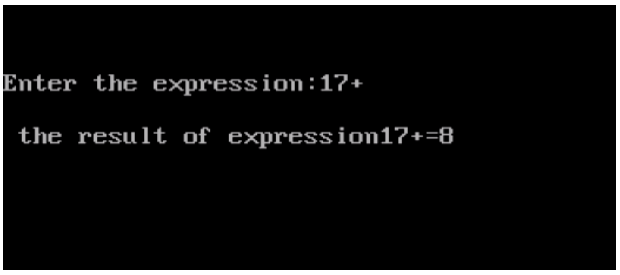
[7] Write a program to evaluate a postfix expression.

```
#include<stdio.h>
#include<conio.h>
#include<ctype.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;
clrscr();
printf("Enter the expression:");
scanf("%s",exp);
e=exp;
while(*e!='\0')
{
if(isdigit(*e))
{
num=*e-(48);
push(num);
```

```
}
else
{
n1=pop();
n2=pop();
switch(*e)
{
case'+':
{
n3=n1+n2;
break;
}
case'-':
{
n3=n2-n1;
break;
}
case'*':
{
n3=n1*n2;
break;
}
case'/':
{
n3=n2/n1;
break;
}
}
push(n3);
}
```

```
e++;
}
printf("\n the result of expression%s=%d\n\n",exp,pop());
getch();
return 0;
}
```

OUTPUT:



[8] write a program to check the operations on stack.

```
#include<stdio.h>
#include<conio.h>
# define size 5
void push();
int pop();
void display();
int top=-1,stk[size];
void main()
{
int ch, item;
clrscr();
while(1)
{
printf("\n1> PUSH\n");
printf("\n2> POP\n");
printf("\n3> DISPLAY\n");
printf("\n4> exit\n");
printf("\nEnter the choice\n");
scanf("%d",&ch);

switch(ch)
{
case 1:
push();
break ;

case 2:
item=pop();
```

```
if(item!=-1)
printf("popped element= %d\n",item);
break;

case 3:
display();
break;

case 4:
//exit(0);
return;
default:
printf("invalid choice\n");
}
}
}
void push(int item)
{
if (top==(size-1))
printf("stack full...stack overflow\n");
else
{
printf("enter the element \n");
scanf("%d",&item);
 top++;
 stk[top]=item;
 }
 }
 void display()
 {
```

```
int i;
i=top;
printf("stack contain \n");
if(i==(-1))
printf("empty stack...stack underflow\n");
else{
while(i!=(-1))
{
printf("%d\t",stk[i]);
i--;

}}}
int pop()
{
int item;
if(top==(-1))
{
printf("empty stack...stack underflow..\n");
return-1;
}
else
{
item=stk[top];
top--;
return (item);
}
}
```

OUTPUT:

```
1> PUSH

2> POP

3> DISPLAY

4> exit

Enter the choice:1
enter the element
1

1> PUSH

2> POP

3> DISPLAY

4> exit

Enter the choice:1
enter the element
2
```

```
Enter the choice:3
stack containt
2        1
1> PUSH

2> POP

3> DISPLAY

4> exit

Enter the choice:2
poped element= 2

1> PUSH

2> POP

3> DISPLAY

4> exit

Enter the choice:
```

```
Enter the choice:2
poped element= 2

1> PUSH

2> POP

3> DISPLAY

4> exit

Enter the choice:2
poped element= 1

1> PUSH

2> POP

3> DISPLAY

4> exit

Enter the choice:
_
```

```
Enter the choice:
3
stack containt
empty stack...stack underflow

1> PUSH

2> POP

3> DISPLAY

4> exit

Enter the choice:
```

[1] Write a program to create singly linked list, insert elements into the list and delete elements from the list.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
Struct node
{
 int info;
 struct node*link;
}
 *start,*nn,*curr,*temp,*prev;
 int menu();
 void sll_insert();
 void sll_delete();
 void sll_display();
 void main()
{
 start=NULL;
 while(1)
{
 switch(menu())
{
 case 1: sll_insert();break;
 case 2: sll_delete();break;
 case 3: sll_display();break;
 case 4: exit(0);
 default: printf("\n Invalid choice...");
}
}
}
```

```
int menu()
{
int ch;
printf("\n operations on singly linked list");
printf("\n1.Insert \n2.Delete \n3.Display \n4.Exit");
printf("\n enter your choice:");
scanf("%d",&ch);
return ch;
}
void sll_insert()
{
int i;
nn=(struct node*)malloc(sizeof(struct node));
if(nn==NULL)
{
printf("\n memory allocation failed...");
return;
}
printf("\n enter the item:");
scanf("%d",&i);
nn->info=i;
nn->link=NULL;
if(start==NULL)
start=curr=nn;
else
{
curr->link=nn;
curr=nn;
}
}
```

```
void sll_delete()
{
int i,flag=0;
printf("\n enter item to be deleted:");
scanf("%d",&i);
temp=start;
while(temp!=NULL)
{
if(temp->info==i)
{
flag=1;
if(temp==start)
start=start->link;
else
prev->link=temp->link;
printf("\n node with information %d is deleted",i);
}
prev=temp;
temp=temp->link;
}
if(flag==0)
printf("\n element not found...");
else
sll_display();
}
void sll_display()
{
if(start==NULL)
{
printf("\n linked list is empty...");
```

```
 return;
}
 temp=start;
 printf("\n linked list elements:");
 while(temp!=NULL)
{
 printf("\t %d",temp->info);
 temp=temp->link;
}
}
```

OUTPUT:

```
 operations on singly linked list
1.Insert
2.Delete
3.Display
4.Exit
 enter your choice:1

 enter the item:1

 operations on singly linked list
1.Insert
2.Delete
3.Display
4.Exit
 enter your choice:1

 enter the item:2
```

```
 operations on singly linked list
1.Insert
2.Delete
3.Display
4.Exit
 enter your choice:3

 linked list elements:   1       2
 operations on singly linked list
1.Insert
2.Delete
3.Display
4.Exit
 enter your choice:2

 enter item to be deleted:1

 node with information 1 is deleted
 linked list elements:   2
 operations on singly linked list
1.Insert
2.Delete
3.Display
4.Exit
 enter your choice:2_
```

```
 enter item to be deleted:2

 node with information 2 is deleted
 linked list is empty...
 operations on singly linked list
1.Insert
2.Delete
3.Display
4.Exit
 enter your choice:_
```

[2] Write a program to create a linear queue to insert elements into the list and delete elements from the list, display your list after every insertion and deletion.

```
#include<stdio.h>
#include<conio.h>
void qinsert();
void qdelete();
void qdisplay();
int queue[10],front=0,rear=-1;
int max=5;
void main()
{
 int ch;
 clrscr();
 do
 {
 printf("\n LINEAR QUEUE OPERTATIONS \n");
 printf("1.Insert \n");
 printf("2.delete\n");
 printf("3.display\n");
 printf("4.exit\n");
 printf("enter your choice: ");
 scanf("%d",&ch);
 switch(ch)
 {
  case 1:qinsert();
            break;
  case 2:qdelete();
            break;
  case 3:qdisplay();
```

```
            break;
  case 4:exit(0);
  default: printf("\n WRONG CHOICE");
  }
  }
  While (ch!=4);
}
void qinsert()
{
 int item;
 if(rear==max-1)
 printf("Queue is full");
 else
 {
 printf("enter the value to insert \n");
 scanf("%d",&item);
 rear++;
 queue[rear]=item;
}
}
  void qdelete()
{
 int item;
 if(front==rear+1)
 printf("Queue is empty");
 else
{
 item=queue[front];
 printf("%d is deleted \n",item);
 front++;
```

```
}
}
 void qdisplay()
{
 int item;
 int p=front;
 if(p==rear+1)
 printf("Queue is empty");
 else
{
 printf("\n Queue elements \n");
 while(p<=rear)
{
 printf("%d\t",queue[p]);
 p++;
}
}
}
```

OUTPUT:

```
 LINEAR QUEUE OPERTATIONS
1.Insert
2.delete
3.display
4.exit
enter your choice: 1
enter the value to insert
1

 LINEAR QUEUE OPERTATIONS
1.Insert
2.delete
3.display
4.exit
enter your choice: 1
enter the value to insert
2

 LINEAR QUEUE OPERTATIONS
1.Insert
2.delete
3.display
4.exit
enter your choice: 3
```

```
 Queue elements
1       2
 LINEAR QUEUE OPERTATIONS
1.Insert
2.delete
3.display
4.exit
enter your choice: 2
1 is deleted

 LINEAR QUEUE OPERTATIONS
1.Insert
2.delete
3.display
4.exit
enter your choice: 2
2 is deleted

 LINEAR QUEUE OPERTATIONS
1.Insert
2.delete
3.display
4.exit
enter your choice: _
```

```
enter your choice: 3
Queue is empty
```

[3] Write a program to create a circular queue to insert an elements into the list and delete elements from the list, Display your list after every insertion and deletion.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define n 5
int CQ[n],f=0,r=-1,i;
void cq_insert();
void cq_delete();
void cq_display();
int menu();
void main()
{
clrscr();
while(10)
{
 switch( menu())
 {
  case 1: cq_insert();break;
  case 2: cq_delete();break;
  case 3: cq_display();break;
  case 4: exit(0);
  default : printf("\n invalid choice .....");
  }
 }
}
int menu()
{
 int ch;
```

```
printf("\n circular queue operations");
printf("\n 1.insert  \n 2. delete \n 3. display \n 4. exit");
printf("\n enter your choice:");
scanf("%d",&ch);
return ch;
}
void cq_insert()
{
 int item;
 printf("\n enter item:");
 scanf("%d",&item);
 if (f==(r+1)%n&& r!=-1)
 {
  printf("\n circular queue is full.... ");
  return;
  }
  r=(r+1)%n;
  CQ[r]=item;
  }
  void cq_delete()
  {
   if (f==(r+1)%n && r==-1)
   {
    printf("\n circular queue is empty...");
    return;
    }
    printf("\n item deleted=%d", CQ[f]);
    f=(f+1)%n;
    if (f==(r+1)%n)
    f=0;r=-1;
```

```
}

void cq_display()
{
if(f==(r+1)%n && r==-1)
{
 printf("\n circular queue is empty...");
 return;
 }
 printf("\n circular queue elements:");
 if (f<=r)
 {
 for (i=f;i<=r;i++)
 printf("\t%d",CQ[i]);
 }
 else
 {
  for (i=f;i<n;i++)
  printf("\t%d",CQ[i]);
  for (i=0;i<=r;i++)
  printf("\t%d",CQ[i]);
  }
  getch();
 }
```

OUTPUT:

```
circular queue operations
1.insert
2. delete
3. display
4. exit
enter your choice:1

enter item:2

circular queue operations
1.insert
2. delete
3. display
4. exit
enter your choice:1

enter item:4

circular queue operations
1.insert
2. delete
3. display
4. exit
enter your choice:3
```

```
circular queue elements:        2       4
circular queue operations
1.insert
2. delete
3. display
4. exit
enter your choice:2

item deleted=2
circular queue operations
1.insert
2. delete
3. display
4. exit
enter your choice:2

item deleted=4
circular queue operations
1.insert
2. delete
3. display
4. exit
enter your choice:3_
```

```
enter your choice:3

circular queue is empty...
```

[4] Write a program to create a Binary tree to insert an element into it and delete elements from it, Display the tree after every insertion and deletion.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct node
{
 int value;
 struct node *left_child, *right_child;
};

 struct node *new_node(int value)
{
 struct node *tmp=(struct node *)malloc(sizeof(struct node));
 tmp->value=value;
 tmp->left_child=tmp->right_child=NULL;
 return tmp;
}


 void delete_tree(struct node *binary_tree)
{
 if (binary_tree)
{
 delete_tree(binary_tree->left_child);
 delete_tree(binary_tree->right_child);
 free(binary_tree);
}
}
```

```
void print(struct node *root_node) // displaying the nodes!
{
if (root_node!=NULL)
{
print(root_node->left_child);
printf("%d \n", root_node->value);
print(root_node->right_child);
}
}

struct node* insert_node(struct node* node, int value) // inserting nodes!
{

if (node==NULL) return new_node(value);
if (value<node->value)
{
node->left_child=insert_node(node->left_child, value);
}
else if (value>node->value)
{
node->right_child=insert_node(node->right_child, value);
}
return node;
}

void search(struct node** cur, int item, struct node** parent)
{
while (cur!=NULL&&(*cur)->value!=item)
{
*parent=*cur;
```

```
 if (item<(*cur)->value)
  *cur=(*cur)->left_child;
 else
 *cur=(*cur)->right_child;
 }
}

 void deletion(struct node* root, int item)
{
 struct node* parent=NULL;
 struct node* cur=root;
 struct node* child=NULL;

 search(&cur, item, &parent);

 if (cur==NULL)
{
 return;
}

 if (cur->left_child==NULL&&cur->right_child==NULL)
{
 if (cur!=root)
{
 if (parent->left_child==cur)
 parent->left_child=NULL;
 else
 parent->right_child=NULL;
}
 else
```

```
 root=NULL;

 free(cur);
}
 else if (cur->left_child && cur->right_child)
{
 struct node* succ=cur->right_child;
 int val=succ->value;
 deletion(root, succ->value);
 cur->value=val;
}
 else
{
 child=(cur->left_child)? cur->left_child: cur->right_child;
 if(cur!=root)
{
 if(cur==parent->left_child)
 parent->left_child=child;
 else
 parent->right_child=child;
}
 else
{
 root=child;
}
 free(cur);
}
}

 void main()
```

```
{
 struct node *root_node=NULL,*node=NULL;
 root_node=insert_node(root_node, 10);
 insert_node(root_node,30);
 insert_node(root_node,25);
 insert_node(root_node,36);
 insert_node(root_node,56);
 insert_node(root_node,78);
 clrscr();
 printf("The elements of the tree traversed in inorder way:\n");
 print(root_node);
 deletion(root_node, 36);
 printf("The elements of the tree after deletion traversed in inorder way:\n");
 print(root_node);
 getch();
}
```

OUTPUT:

```
The elements of the tree traversed in inorder way:
10
25
30
36
56
78
The elements of the tree after deletion traversed in inorder way:
10
25
30
56
78
```

[5] Write a program to create a Binary search tree to insert an elements into it and perform inorder, preorder and postorder traversal.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct Node
{
  int data;
  struct Node *left, *right;
};

 struct Node* newNode(int data)
{
 struct Node* temp;
 temp=(struct Node*)malloc(sizeof(struct Node));
 temp->data=data;
 temp->left = temp->right = NULL;
 return temp;
}

 void printPostorder(struct Node* node)
{
 if (node==NULL)
 return;

 printPostorder(node->left);
 printPostorder(node->right);
 printf("%d ",node->data);
}
```

```
void printInorder(struct Node* node)
{
if(node==NULL)
return;

printInorder(node->left);
printf("%d ", node->data);
printInorder(node->right);
}

void printPreorder(struct Node* node)
{
if(node==NULL)
return;

printf("%d ", node->data);
printPreorder(node->left);
printPreorder(node->right);
}

int main()
{
 struct Node* root = newNode(3);
 root->left = newNode(1);
 root->right = newNode(4);
 root->left->left = newNode(0);
 root->left->right = newNode(2);
 clrscr();
 printf("\nPreorder traversal of binary tree is:\n");
 printPreorder(root);
```

```
  printf("\nInorder traversal of binary tree is:\n");
  printInorder(root);

  printf("\nPostorder traversal of binary tree is:\n");
  printPostorder(root);

  return 0;
  getch();
}
```

OUTPUT:

```
Preorder traversal of binary tree is:
3 1 0 2 4
Inorder traversal of binary tree is:
0 1 2 3 4
Postorder traversal of binary tree is:
0 2 1 4 3
```

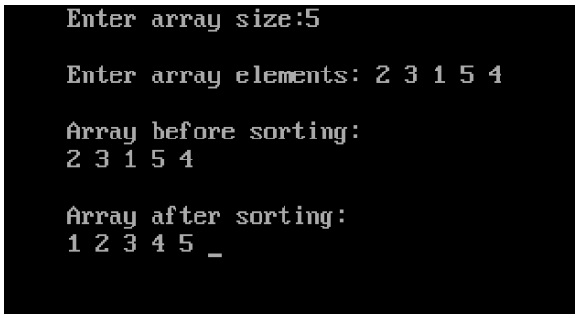
[6] Write a program to implement Heap sort in C.

```
#include<stdio.h>
#include<conio.h>
void heapify(int*,int, int);
void heapsort(int*, int);
void print_array(int*, int);
void main()
{
 int i,arr[100],n;
 clrscr();
 printf("Enter array size:");
 scanf("%d", &n);
 printf("\nEnter array elements: ");
 for(i=0;i<n;i++)
 scanf("%d", &arr[i]);
 printf("\nArray before sorting:\n");
 print_array(arr, n);
 heapsort(arr, n);
 printf("\n\nArray after sorting:\n");
 print_array(arr, n);
}
void heapsort(int* arr, int n)
{
 int i;
 for(i=n/2-1;i>=0;i--)
{
 heapify(arr, n, i);
}
 for(i=n-1;i>=0;i--)
```

```
{
 int temp=arr[i];
 arr[i]=arr[0];
 arr[0]=temp;
 heapify(arr, i, 0);
}
}
void heapify(int* arr, int n, int i)
{
 int largest=i;
 int left=2*i+1;
 int right=2*i+2;
 if(left<n&&arr[left]>arr[largest])
{
 largest=left;
}
 if(right<n&&arr[right]>arr[largest])
{
 largest=right;
}
 if(largest!=i)
{
 int temp=arr[i];
 arr[i]=arr[largest];
 arr[largest]=temp;
 heapify(arr, n, largest);
}
}
 void print_array(int* arr, int n)
{
```

```
 int i;
 for(i=0;i<n;i++)
{
 printf("%d ",arr[i]);
}
 getch();
}
```

OUTPUT:



[7] Write a program to implement the string function for:

[1] Finding length of a string.

[2] Concatenate 2 strings.

[3] Extract a sub string from a given string.

[4] Replace a character in a string.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
int stringlength()
{
char s[100];
printf("lnput of the string:\n");
scanf("%s",&s);
printf("length of the string %s is %d \n",s,strlen(s));
return 0;
}
int concatenate()
{
char destination[100],source[100];
printf("input first string:\n");
scanf("%s",&destination);
printf("input second string: \n");
scanf("%s",&source);
strcat(destination,source);
printf("concatenated string:%s \n",destination);
return 0;
}
 int substring()
```

```
{
char string[100],sub[100];
int position,length,c=0;
printf("input a string: \n");
scanf("%s",string);
printf("enter the position and length of substring: \n");
scanf("%d%d",&position,&length);
while(c<length)
{
sub[c]=string[position+c-1];
c++;
}
sub[c]='\0';
printf("required substring is \'%s\'\n",sub);
return 0;
}
    void charactereplace()
{
char string[100],ch,replacech;
int i;
printf("input string:\n");
scanf("%s",&string);
printf("input character to find:\n ");
 ch=getche();
printf("input character to replace:");
replacech=getche();
for(i=0;i<strlen(string);i++)
{
if(string[i]==ch)
string[i]=replacech;
```

```
}
printf("\nstring after replacing:\n");
printf("%s\n",string);
}
int main()
{
int choice;
clrscr();
while(1)
{
printf("1. string length \n 2.string concatenation \n  3. substring extracting \n 4. character
replacement \n");
printf("enter your choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:
stringlength();
break;
case 2:
concatenate();
break;
case 3:
substring();
break;
case 4:
characterreplace();
break;
default:
exit(0);
}
```

```
}

}
```

OUTPUT:

```
1.string length
2.string concatenation
3.substring extracting
4.character replacement
enter your choice:1
Input of the string:
indian_academy
length of the string indian_academy is 14

1.string length
2.string concatenation
3.substring extracting
4.character replacement
enter your choice:2
input first string:
hi
input second string:
hello
concatenated string:hihello

1.string length
2.string concatenation
3.substring extracting
4.character replacement
enter your choice:_
```

```
enter your choice:3
input a string:
indian
enter the position and length of substring:
6
6
required substring is 'n'

1.string length
2.string concatenation
3.substring extracting
4.character replacement
enter your choice:4
input string:
apademy
input character to find:
 pinput character to replace:c
string after replacing:
academy

1.string length
2.string concatenation
3.substring extracting
4.character replacement
enter your choice:_
```