OUTPUT

```
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit 1
Enter data : 5
queue:5
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit 1
Enter data : 2
queue:5, 2
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit 1
Enter data : 6
queue:5, 2, 6
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit 2
Data removed: 5
queue:2, 6
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit 1
Enter data : 9
queue:2, 6, 9
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit
3
queue:9, 6, 2
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit 1
Enter data : 1
queue:9, 6, 2, 1
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit 3
queue:1, 2, 6, 9
Enter
1.enqueue
2.dequeue
3.reverse queue
4.exit
4
```

OUTPUT

```
enter the first polnomial
Enter the number of terms:3
Enter the coefficient  for term 1 :5
Enter the exponent for the term 1 :3
Enter the coefficient  for term 2 :2
Enter the exponent for the term 2 :2
Enter the coefficient  for term 3 :1
Enter the exponent for the term 3 :0
enter the second polnomial
Enter the number of terms:3
Enter the coefficient  for term 1 :3
Enter the exponent for the term 1 :3
Enter the coefficient  for term 2 :1
Enter the exponent for the term 2 :1
Enter the coefficient  for term 3 :5
Enter the exponent for the term 3 :0

FIRST POLYNOMIAL    : (5x^3)+(2x^2)+(1x^0)

SECOND POLYNOMIAL   : (3x^3)+(1x^1)+(5x^0)

RESULTANT POLYNOMIAL:(15x^6)+(6x^5)+(5x^4)+(30x^3)+(10x^2)+(1x^1)+(5x^0)
```

OUTPUT

```
enter the first polnomial
Enter the number of terms:2
Enter the coefficient  for term 1 :9
Enter the exponent for the term 1 :3
Enter the coefficient  for term 2 :5
Enter the exponent for the term 2 :2
enter the second polnomial
Enter the number of terms:2
Enter the coefficient  for term 1 :10
Enter the exponent for the term 1 :2
Enter the coefficient  for term 2 :20
Enter the exponent for the term 2 :1

FIRST POLYNOMIAL    : (9.0x^3)+(5.0x^2)

SECOND POLYNOMIAL   : (10.0x^2)+(20.0x^1)

RESULTANT POLYNOMIAL:(9.0x^3)+(15.0x^2)+(20.0x^1)
```

PROGRAM

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

struct Node{
	char data;
	struct Node* prev;
	struct Node* next;
};

struct Node *head=NULL,*tail=NULL;

void todoubly(char str[]){
	for(int i=0;str[i]!='\0';i++){
		struct Node* new_node = malloc(sizeof (struct Node));
		new_node->data = str[i];
		new_node->prev = tail;new_node->next = NULL;
		if(!head)
			head = new_node;
		else
			tail->next = new_node;
		tail = new_node;
	}
}

void check_palindrome(){
	int flag=1;
	struct Node *h_temp=head,*t_temp=tail;
	if(head){
		while(h_temp!=t_temp){
			if(h_temp->data != t_temp->data){
				flag=0;
				break;
			}
			h_temp = h_temp->next;
			t_temp = t_temp->prev;
		}
	}if(flag)
		printf("String is palindrome\n");
	else
		printf("String is not palindrome\n");
}
```

```
int main(void){
	char str[50];
	printf("Enter a string\n");
	scanf("%s",str);
	todoubly(str);
	check_palindrome();
}
```

OUTPUT

```
Enter a string :malayalam
String is palindrome

Enter a string :apple
String is not palindrome
```

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

struct node{
        int data;
        struct node *left,*right;
}*root=NULL;

struct node *createNode(int data){
        struct node *node=malloc(sizeof(struct node));
          node->data = data;
          node->left =node->right = NULL;
          return node;
}

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);
}

struct node *inorderSuccessor(struct node *node){
        struct node *current = node->right;
        while (current->left != NULL)
                current = current->left;
        return current;
}
```

```
struct node *deleteNode(struct node *root,int data){
    if (root == NULL)
        return root;

    if (data < root->data)
        root->left = deleteNode(root->left, data);
    else if (data > root->data)
        root->right = deleteNode(root->right, data);
    else{
        if (root->left == NULL){
            struct node *temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL){
            struct node *temp = root->left;
            free(root);
            return temp;
        }
        struct node *temp = inorderSuccessor(root);
        root->data = temp->data;
        root->right = deleteNode(root->right,temp->data);
    }
     return root;
}


struct node *insertNode(struct node *node,int data){
     if (node == NULL)
        return createNode(data);
    if (data < node->data)
        node->left = insertNode(node->left,data);
    else if (data > node->data)
        node->right = insertNode(node->right,data);
    return node;
}

int main(){
  int i,n,choice,data;
  while(1){
    printf("\nBINARY_SEARCH_TREE\n1.Insert Node\n"
    "2.Inorder   Traversal\n3.Preorder  Traversal"
    "\n4.Postorder Traversal\n5.Delete a Node\n"
    "6.Exit\nEnter your choice :");
    scanf("%d",&choice);
    switch(choice){
        case 1 :
            printf("Enter the number of nodes : ");
```

```
                        scanf("%d", &n);
                        printf("Input the nodes : \n");
                        scanf("%d", &data);
                        root = insertNode(root, data);
                        for (i = 1; i < n; i++){
                                scanf("%d", &data);
                                insertNode(root, data);
                        }
                        break;
                case 2 :
                        printInorder(root);
                        break;
                case 3 :
                        printPreorder(root);
                        break;
                case 4 :
                        printPostorder(root);
                        break;
                case 5 :
                        printf("Enter the node to be deleted : ");
                        scanf("%d", &data);
                        deleteNode(root, data);
                        printf("Inorder traversal after deletion:\n");
                        printInorder(root); break;
                case 6 :
                         exit(1);
                default:
                        printf("Invalid Input !Try again..");
        }
    }
    return 0;
}
```

OUTPUT

```
BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :1
Enter the number of nodes : 9
Input the nodes :
75 50 100 30 65 68 67 70 20

BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :2
20 30 50 65 67 68 70 75 100
BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :3
75 50 30 20 65 68 67 70 100
BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :4
20 30 67 70 68 65 50 100 75
BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :5
Enter the node to be deleted : 20
Inorder traversal after deletion :
 30 50 65 67 68 70 75 100
BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :5
Enter the node to be deleted : 65
Inorder traversal after deletion :
 30 50 67 68 70 75 100
BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :1
Enter the number of nodes : 2
Input the nodes :
10 5

BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :2
5 10 30 50 67 68 70 75 100
BINARY_SEARCH_TREE
1.Insert Node
2.Inorder   Traversal
3.Preorder  Traversal
4.Postorder Traversal
5.Delete a Node
6.Exit
Enter your choice :6
```

PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
int graph[20][20],visited[20],q[20];
int n, front=-1,rear=-1;

void bfs(int start){
    front=rear=0;
    q[rear]=start;
    visited[start]=1;
    printf("%d -> ",start);
    while(front<=rear){
        for(int i=0;i<n;i++)
            if(graph[start][i] && !visited[i]){
                q[++rear]=i;
                visited[i]=1;
                printf("%d -> ",i);
            }
        start=q[++front];
    }
}

void dfs(int start) {
  printf("%d -> ", start);
  visited[start] = 1;
  for (int i = 0; i < n; i++)
        if (graph[start][i] && !visited[i])
            dfs(i);
}

int main() {
  int start;
  printf("Enter the number of vertex: ");
  scanf("%d", &n);
  printf("Enter the adjacency matrix: \n");
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
         scanf("%d", &graph[i][j]);
  printf("Enter the start vertex: ");
  scanf("%d", &start);
  printf("\nDFS : ");
  dfs(start);
  for(int i=0;i<n;i++)
      visited[i]=0;
  printf("\nBFS : ");
  bfs(start);
}
```

OUTPUT

```
Enter the number of vertex: 8
Enter the adjacency matrix:
0 1 0 0 0 0 1 0
1 0 1 0 1 0 0 0
0 1 0 1 1 0 0 0
0 0 1 0 0 0 0 0
0 1 1 0 0 1 0 0
0 0 0 0 1 0 0 1
1 0 0 0 0 0 0 1
0 0 0 0 0 1 1 0
Enter the start vertex: 0

DFS : 0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 7 -> 6 ->
BFS : 0 -> 1 -> 6 -> 2 -> 4 -> 7 -> 3 -> 5 ->
```

```
Enter the number of vertex: 5
Enter the adjacency matrix:
0 0 0 1 1
0 0 1 0 1
0 1 0 1 0
1 0 1 0 0
1 1 0 0 0
Enter the start vertex: 0

DFS : 0 -> 3 -> 2 -> 1 -> 4 ->
BFS : 0 -> 3 -> 4 -> 2 -> 1 ->
```

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
int arr[50],size;


void printArray(){
        printf("Sorted array:");
        for(int i=0 ;i<size;i++)
                printf("%d ", *(arr+i) );
}

void readArray(){
        printf("Enter the limit: ");
        scanf("%d", &size);
        printf("Enter the integers: ");
        for(int i=0;i<size;i++)
                scanf("%d", arr+i );
}



void swap(int *arr,int i,int j){
        int temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
}



void bubbleSort(int *arr, int n) {
        for (int i = 0; i < n - 1; i++)
                for (int j = 0; j < n - i - 1; j++)
                        if (arr[j] > arr[j + 1])
                                swap(arr,j,j+1);
}


void insertionSort(int *arr, int n){
         for (int i = 1; i < n; i++){
                int key = arr[i];
                int j = i - 1;
                while (j >= 0 && arr[j] > key)
                        arr[j-- + 1] = arr[j];
                arr[j + 1] = key;
        }
}
```

```
void selectionSort(int* arr, int size) {
        for (int i = 0; i < size - 1; i++) {
                int min_index = i;
                for (int j = i + 1; j < size; j++)
                        if (arr[j] < arr[min_index])
                                min_index = j;
                if(i!=min_index)
                swap(arr,i,min_index);
        }
}

int partition(int *arr, int left, int right) {
        int i = left + 1,j;
        int pivot = arr[left];
        int tmp;
        for (j = left + 1; j <= right; j++)
                if (arr[j] < pivot) {
                swap(arr,i,j);
                i++;
        }
        swap(arr,left,i-1);
        return i - 1;
}

void quickSort(int *arr, int left, int right) {
        if (left >= right)
                return;
        int pivot_index = partition(arr, left, right);
        quickSort(arr, left, pivot_index - 1);
        quickSort(arr, pivot_index + 1, right);
}

void merge(int *arr, int left, int mid, int right){
        int i = left;
        int j = mid + 1;
        int temp[right - left + 1];
        int k = 0;
        while (i <= mid && j <= right)
                if (arr[i] <= arr[j])
                        temp[k++] = arr[i++];
                else
                        temp[k++] = arr[j++];
        while (i <= mid)
                temp[k++] = arr[i++];
        while (j <= right)
                temp[k++] = arr[j++];
        for (i = left, k = 0; i <= right; i++, k++)
                arr[i] = temp[k];
}


void mergeSort(int *arr, int left, int right){
         if ( left >= right )
                return;
        int mid = ( left + right ) / 2;
        mergeSort( arr , left , mid );
        mergeSort( arr, mid + 1, right);
        merge( arr, left, mid, right);
}

int main(){
        int choice;
        while(1){
                printf("\nEnter \n1.Bubble Sort\n2.Insertion Sort"
                "\n3.Selection Sort\n4.Quick Sort\n5.Merge Sort\n6.Exit\n:");
                scanf("%d",&choice);
                if(choice == 6)
                         exit(1);
                readArray();
                switch(choice){
                case 1 : bubbleSort(arr,size);
                         break;
                case 2 : insertionSort(arr,size);
                         break;
                case 3 : selectionSort(arr,size);
                         break;
                case 4 : quickSort( arr,0,size-1);
                         break;
                case 5 : mergeSort(arr,0,size-1);
                         break;
         }
    printArray();
  }
}
```

OUTPUT

```
Enter
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Quick Sort
5.Merge Sort
6.Exit
:1
Enter the limit: 5
Enter the integers: 7 1 6 2 9
Sorted array:1 2 6 7 9
Enter
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Quick Sort
5.Merge Sort
6.Exit
:2
Enter the limit: 5
Enter the integers: 6 2 3 1 7
Sorted array:1 2 3 6 7
Enter
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Quick Sort
5.Merge Sort
6.Exit
:3
Enter the limit: 5
Enter the integers: 1 0 2 5 4
Sorted array:0 1 2 4 5
Enter
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Quick Sort
5.Merge Sort
6.Exit
:4
Enter the limit: 5
Enter the integers: 3 0 1 7 2
Sorted array:0 1 2 3 7
Enter
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Quick Sort
5.Merge Sort
6.Exit
:5
Enter the limit: 5
Enter the integers: 11 0 2 3 1
Sorted array:0 1 2 3 11
Enter
1.Bubble Sort
2.Insertion Sort
3.Selection Sort
4.Quick Sort
5.Merge Sort
6.Exit
:6
```