|  |
| --- |
| **Q2.Implement above insert and removal of elements in array using function pointers. Provide options of insert, delete, display and end of input operations. Repeat until user wants to end the process.**  #include<stdio.h>  void (\*array\_insert)(int\*,int,int,int\*);  void (\*array\_delete)(int\*,int,int\*);  void arrayinsert(int array[],int position, int data, int \*length)  {  int index;  for(index=(\*length)-1;index>=1;index--)  array[index+1]=array[index];  array[position-1]=data;  \*length=(\*length)+1;  }  void arraydelete(int array[],int position, int \*length)  {  int index;    for(index=position-1;index<(\*length)-1;index++)  array[index]=array[index+1];  \*length=(\*length)-1;  }  void display(int array[],int length)  {  int index;  for(index=0;index<length;index++)  printf("array[%d]=%d\n",index,array[index]);  printf("\n");  } |
| **Output:**  **Enter array length :5**  **Enter array element:**  **array[0]=1**  **array[1]=2**  **array[2]=3**  **array[3]=4**  **array[4]=5**  **Enter your choice:**  **1.Insert 2. Delete 3. Display 4. Exit**  **1**  **Enter array position and element to insert:2**  **5**  **Element inserted successfully**  **Enter your choice:**  **1.Insert 2. Delete 3. Display 4. Exit**  **3**  **array[0]=1**  **array[1]=5**  **array[2]=2**  **array[3]=3**  **array[4]=4**  **array[5]=5**  **Enter your choice:**  **1.Insert 2. Delete 3. Display 4. Exit**  **2**  **Enter position to delete:3**  **Element deleted successfully**  **Enter your choice:**  **1.Insert 2. Delete 3. Display 4. Exit**  **3**  **array[0]=1**  **array[1]=5**  **array[2]=3**  **array[3]=4**  **array[4]=5**  **Enter your choice:**  **1.Insert 2. Delete 3. Display 4. Exit**  **4** |
| **Q3. Write a function to accept argument as function pointer. Include main program to test it.**  **Test with prime number function implemented above.**  #include<stdio.h>  void function(int,void(\*fp)(int));  void function1(int);  void (\*p)(int);  int main()  {  int number;  p=function1;  printf("\n Enter the number to check prime:");  scanf("%d",&number);  function(number,p);  return 0;  }  void function(int number,void(\*fp)(int))  {  (\*fp)(number);  }  void function1(int number)  {  int loop,flag=0;  for(loop=2;loop<number/2;loop++)  {  if(number%loop==0)  {  flag=1;  break;  }  }  if(flag==0)  printf("\n Prime \n");  else  printf("\n Not a Prime \n");  } |
| **Output:**  Enter the number to check prime:5  Prime  Enter the number to check prime:8  Not a Prime |
| **Q5. Take an array of few elements. Point first pointer to first element and last pointer to last**  **element. Subtract last and first pointers from each other and test the results.**  #include<stdio.h>  int main()  {  int array[100], index, length,\*first,\*last;  printf("\n Enter the array length:");  scanf("%d",&length);  printf("\n Enter the array elements: \n");  for(index=0;index<length;index++)  {  printf(" array[%d]=",index);  scanf("%d",&array[index]);  }  first=&array[0];  last=&array[length-1];  printf("\n pointer substraction= %d value substraction=%d \n",last-first,\*last - \*first);  return 0;  } |
| **Output:**  **Enter the array length:5**  **Enter the array elements:**  **array[0]=1**  **array[1]=2**  **array[2]=3**  **array[3]=4**  **array[4]=5**  **pointer substraction= 4 value substraction=4** |
| **Q12. Write a program to demonstrate pointer to an array. Initialize values with 3 X 3 matrix,**  **access row 1-col 1, row2-col1, row3-col1 using pointer to an array**  #include<stdio.h>  int main()  {  int array[3][3],row=3,col=3,index1,index2,(\*p)[3];    printf("\n Enter array elements:\n");  for(index1=0;index1<row;index1++)  {  for(index2=0;index2<col;index2++)  {  printf("array[%d][%d]=",index1,index2);  scanf("%d",&array[index1][index2]);  }  }    p=&array;  index2=0;  printf("\n\n");    for(index1=0;index2<row;index2++)  {  printf("array[%d][%d]=%d\n",index1,index2,array[index1][index2]);  }  } |
| **Output:**  **Enter array elements:**  **array[0][0]=1**  **array[0][1]=2**  **array[0][2]=3**  **array[1][0]=4**  **array[1][1]=5**  **array[1][2]=6**  **array[2][0]=7**  **array[2][1]=8**  **array[2][2]=9**  **array[0][0]=1**  **array[0][1]=2**  **array[0][2]=3** |
| **Q14.Write a program to demonstrate malloc() and free() functionality. Use allocated memory**  **to store multiple of number, say, num. (Accept num and number of elements from user,**  **repeat the process until user enters 0 or -1 value for num)**  #include<stdio.h>  #include<stdlib.h>  #define SIZE 100  int main()  {  int index1=-1,index2;  int \*ptr=(int \*)malloc(SIZE\*sizeof(int));  if(ptr==NULL)  {  printf("Fail to allocate memory \n");  exit(0);  }  printf("\n Enter the elements:\n");  do  {  index1++;  printf("ptr[%d]=",index1);  scanf("%d",&ptr[index1]);  }while(ptr[index1]>0);  for(index2=0;index2<index1;index2++)  {  printf("array[%d]=%d\n",index2,\*(ptr + index2));  }  free(ptr);  } |
| **Output:**  **Enter the elements:**  **ptr[0]=1**  **ptr[1]=2**  **ptr[2]=3**  **ptr[3]=4**  **ptr[4]=5**  **ptr[5]=0**  **array[0]=1**  **array[1]=2**  **array[2]=3**  **array[3]=4**  **array[4]=5** |